A PROBABILISTIC REASONING AND LEARNING SYSTEM

BASED ON BAYESIAN BELIEF NETWORKS

ZHIYUAN LUO

A Thesis Submitted in Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

HERIOT-WATT UNIVERSITY

Department of Computer Science

MAY 1992

DECLARATION

I hereby declare that I have personally composed this thesis and the work herein was my own except where due acknowledgement is made. The material in this thesis has not been submitted to this or any other university for a degree. However, the early versions of some material presented in this thesis have been published or submitted for publication, which are included in the references list.

Signed:

Dated:

To my mother

Huang Jindi

CONTENTS

List of Tables

Abbreviations

Abbre.	Full
AI	Artificial Intelligence
BBN	Bayesian Belief Network
BI	The Bayesian Inference
CF	The Certainty Factor Model
CG	Conditional Gaussian
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
DS	The Dempster-Shafer Theory
ID	Influence Diagram
IM	Information Measure
KBS	Knowledge Base System
KLD	Kullback-Leibler Distance
L-S	Lauritzen and Spiegelhalter
MWSP	Maximum Weight Spanning Tree
PRESS	Probabilistic Reasoning Expert Shell System
QPN	Qualitative Probabilistic Network

List of Illustrations

ACKNOWLEDGEMENTS

I am most grateful to my supervisor, Dr. Alex Gammerman, for his stimulating guidance, helpful discussion, and steady support throughout my study.

I would like to thank Greg Michaelson, Mark Brewer and Cliff Thomas for their patience on reading and correcting earlier drafts of this thesis, and also for their helpful suggestions.

Thanks are also due to the Chinese Education Commission and the British Council for providing me a Technical Cooperation Training award that has enabled me to complete this research.

I would like to express my thanks to the Department of Computer Science at Heriot-Watt University for the cooperation and support by all members of staff, and the fellow Ph.D students.

Finally, I am indebted to my parents and sister for their constant support and encourage during the past years.

ABSTRACT

Various probabilistic reasoning mechanisms have been developed with almost all of them suffering two points of difficulty. The first is how to construct and encode expertise into a coherent probabilistic form, namely a representational problem. The second, the inferential problem, is how to develop a computationally manageable algorithm for probabilistic reasoning and decision making.

Bayesian belief networks, a graphical knowledge representation, seem to overcome the above difficulties. In this thesis, we consider a probabilistic reasoning and learning system based on Bayesian belief networks. A general probabilistic reasoning expert system shell, PRESS, is then developed and implemented to deal with a range of problems, including both pure and mixed probabilistic models. Examples from forensic science and the forecast of a crop yield are used for illustration purposes.

PRESS can be seen to provide a platform for studying and dealing with machine learning issues in Bayesian belief networks. Aspects, such as coping with inexact probabilities and constructing graphical knowledge representations directly from a database are considered, using the system, with applications on a large medical database being made for the learning processes.

Chapter 1

INTRODUCTION

Expert systems are computer programs, intended to help making judgements or give advice to users in a reasonably convincing way and perform a task that is ordinarily carried out by an expert of the specific domain. They allow a computer to use expertise to assist in a variety of problems. Such systems are generally thought to comprise of three main components: a knowledge base, an inference engine, with the addition of an area of working memory to hold information generated and used only while the system is active [1]. The knowledge base holds the facts and domain-specific procedural information which make up the system's "expertise" which is, in turn, a representation of a human expert's knowledge. The inference engine is responsible for inferring information not already in the system. It does this by applying the expertise found in the knowledge structures of the knowledge base to the information supplied, in a controlled manner. Although a mechanism for handling uncertainty is often included as a requirement of an expert system, this is not always necessary as in some areas of expertise it may be that no uncertainty ever affects the data or the derived conclusions. The user interface is probably the area where most variability is found. However, in any expert system it must provide facilities to allow the user to be given an explanation of the system's reasoning in addition to those for basic data input and results output.

In rule-based expert systems, the knowledge base is in the form of IF "premises" THEN "conclusions" rules, invariably with a weight. An inference procedure typically consists of rule gathering and rule firing. The former is to gather all the relevant rules in a proper order, which is often referred to as a way of specifying or controlling how inferences should be performed. The latter is to apply or execute those rules depending on the kinds of inference to be conducted, which is usually referred to as a way of making inference. Rule gathering can be forward chaining or backward chaining. Forward chaining starts with some basic propositions, and tries to gather those rules in which these propositions appear in their premises. The conclusions of those rules being selected are then used to gather other relevant rules. This process continues until no such rules can be found. Backward chaining is used to chain all the relevant rules in a similar way except that it starts from a goal to be established, and chains rules by always

forward chaining. Thus, forward chaining or backward chaining is used to indicate how relevant information can be collected and made ready for inference. It is noted that forward or backward chaining are commonly used in expert systems [1,2]. For example, backward chaining is employed in the MYCIN system [1], while INTERNIST [3] initially adopts forward chaining but later changes to backward chaining when an initial set of hypotheses is established.

Problem solving and decision making by human experts are often carried out where information concerning the problem is uncertain. There are different sources of uncertainty. For example, due to the complexity of the world, any event seems to be affected or related to numerous other events and relationships between events are often stochastical. However, humans can use this uncertain knowledge effectively to draw conclusions, because although events may be uncertain, in the sense that it is not known whether they are true or false or whether they will occur or not, some events are more likely to occur than others. Specifically we want to attach to any uncertain event, a measure that describes that uncertainty, thereby, we can manipulate these measures and draw conclusions in the light of evidence. Expert systems are characterised by their aim of simulating the behaviour of a human expert rather than the characteristics of the domain. To reach similar conclusions, the expert system must be designed to have the capability to reason under uncertainty.

Although there are many varied concepts of uncertainty within expert systems, it is recognised that it is crucial to represent and deal with uncertain knowledge. The capture and manipulation of uncertain knowledge is fundamentally different from the corresponding tasks associated with knowledge held with certainty. Many attempts have been made to treat uncertainty in the development of expert systems [2,4,5,6,7,8,9,10,11]. Various approaches have been proposed in dealing with knowledge representation and uncertain reasoning [12,13,14,15,16]. Basically, these approaches can be divided into two main groups, depending on whether a system for reasoning uses numbers to represent uncertainty or does not [17,18].

(1) Non-numerical approaches

These consider uncertainty as a kind of knowledge, use symbolic reasoning and avoid any numerical assessments. The most notable examples are the use of nonmonotonic logic [19] and the theory of endorsements [9].

(2) Numerical approaches

In contrast to non-numerical approaches, numerical approaches make use of numbers to represent strength of belief, and of mathematics to manipulate these numbers. They include, for example, possibility theory [6,20], the certainty factor model [1,21], Bayesian inference [2,4,22] and the Dempster-Shafer (DS) theory of evidence [13,23].

Rule-based expert systems employing numerical approaches, developed in the 1970s, paid relatively little attention to the theory of probability. They used numbers to score the strength of evidence, but they did not use ideas from probability theory to organise inference. The primary organising idea was still symbolic logic. Therefore, there are some problems associated with the development of such rule-based expert systems. For example, lack of expressiveness of the knowledge representation language and ignoring of dependency of evidence. In order to cope with these problems in practice, many Bayesian inference systems had to make some unrealistic assumptions, for example, the conditional independence assumption used in PROSPECTOR [2]. To follow a normative probabilistic approach in uncertainty management, researchers of expert systems essentially had to make a choice between oversimplified and computationally intractable domain models.

Bayesian belief networks, a representation that has recently attracted a considerable interest among artificial intelligence researchers, provide a framework for building a coherent probabilistic model of a given domain [24,25,26,27,28]. Bayesian belief networks capture both qualitative and quantitative domain knowledge. Qualitative knowledge is represented by a directed acyclic graph, where nodes represent random variables and arrows represent dependence among certain variables. Quantitative knowledge consists of a set of conditional probability tables attached to each node, which specify probability distribution of the node given all possible combinations of values its parent nodes may take. Different probabilistic reasoning algorithms have been developed based on this representation [25,28].

In addition to the observation of difficulties in the development of probabilistic expert systems, this thesis considers a probabilistic reasoning and learning system based on Bayesian belief networks. Consequently, the objectives of the thesis are

1: to study available statistical algorithms;

2: to develop computational algorithms and design a probabilistic reasoning system;

3: to experiment with different algorithms (exact and approximate) using the developed system;

4: to extend the system to accommodate different learning techniques (parameter and structure learning);

5: to apply the system to a number of examples and evaluate the performance of the system.

As a result, a general probabilistic reasoning expert system shell is developed and implemented to deal with a range of problems, including both pure and mixed probabilistic models. Different aspects of learning in Bayesian belief networks are examined in the course of this development.

The organisation of the thesis is as follows. Probability related uncertain reasoning mechanisms are reviewed in chapter 2. As a result, the lack of expressiveness in the knowledge representation and the intractability of the exhaustive probabilistic calculus are identified as the main difficulties in the design and implementation of probabilistic expert systems. To overcome the difficulties, a graphical knowledge representation, namely Bayesian belief networks, is introduced. Three notable probabilistic reasoning mechanisms, Pearl's message-passing algorithm [25,29], Lauritzen-Spiegelhalter's clique tree algorithm [28,30,31] and a stochastic simulation algorithm [32], are considered. The systematic study and analysis provide a basis for the design and implementation of a generic probabilistic reasoning expert system shell in the next chapter.

Chapter 3 is concerned with the design and implementation of a probabilistic reasoning expert system shell (PRESS) [33]. A basic open computational architecture is established. It consists of four main parts: model construction, preprocessor, evidence propagation and control facilities. The system is evaluated using an example from forensic science. PRESS provides a framework for studying various issues in probabilistic reasoning and learning using Bayesian belief networks, for example, parameter learning and structure learning.

Chapter 4 presents an extension of PRESS to mixed probabilistic models, where some variables are discrete and some continuous [34], based on the open architecture developed in chapter 3. Due to the asymmetry between discrete and continuous variables, the Bayesian belief network and computational model have to satisfy certain constraints [34]. The differences between pure discrete models and mixed models are addressed. This extension enhances the applicability of PRESS and provides a natural way of expressing measures, which results in models in better accordance with reality. An example from agricultural forecasting is used to illustrate the extended PRESS [35].

Chapter 5 deals with learning in Bayesian belief networks. Two related tasks, parameter learning and structure learning, are studied [29,36]. PRESS also provides us with a basis for accommodating these learning techniques. Coping with imprecise probability specification as a data base of cases accumulates and constructing Bayesian belief networks directly from a database are considered using PRESS. Experiments with these techniques on a simulated database and a large medical database [37] have been made and evaluation conducted. The results suggest that the development of learning techniques can enhance the performance of the system. The problems associated with the learning techniques are discussed.

In chapter 6, the underlying problems of probabilistic reasoning in knowledge based systems are reviewed. There is a discussion of some remaining problems together with suggestions for further research work.

Chapter 2

PROBABILITY RELATED REASONING MECHANISMS

As discussed in chapter 1, the development of reasoning mechanisms to cope dynamically with "uncertain" situations is a central task in building expert systems. To provide an effective way of coping with uncertain information, a variety of reasoning mechanisms have been proposed [1,2,7,25,26,27,28,38]. In this chapter, we consider probability related mechanisms in the treatment of uncertainty in expert systems.

First, elementary conceptions of probability theory will be reviewed. They include frequent probability, subjective probability and Bayes' theorem. In many expert systems, estimates are made of probabilities and these estimates are continually updated as new pieces of evidence become available. Bayes' theorem provides a convenient way for doing this. The application of Bayes' theorem to real problems will then be discussed.

Next, Bayesian systems, especially the PROSPECTOR model [2,12] and the G&T model [38], are described and discussed. We are concerned here with how uncertain information is represented and how it is manipulated. The two models will be introduced with respect to these two aspects, along with critical judgements on them.

Finally, the study has enabled us to identify the problems to be solved and the work to be carried out in probabilistic reasoning systems. The lack of expressiveness in the knowledge representation and the intractability of the exhaustive probabilistic calculus are the main difficulties in the design and implementation of probabilistic expert systems. To overcome the difficulties a graphical knowledge representation, namely Bayesian belief networks, is introduced. Three reasoning algorithms using Bayesian belief networks are carefully studied. Computational algorithms are developed and investigated.

2.1. Probability Theory

A probability is a numerical measure of how likely it is (or at least, how likely we think it is) that a certain event will happen based on the information held by a person at some time. Probability theory is a set of axioms that defines measures of belief in events and describes how such measures can be made consistent or combined to infer measure of belief in related events.

2.1.1. Sample Space and Frequency Interpretation of Probability

As an alternative, probabilities can defined using a sample space. Every conceivable event is represented by a point in the sample space. Each point in the sample space is associated with a number which is the probability that the event will occur. The sum of the probabilities in the space must be 1. The probability of an event A is the frequency with which the event happens in a long series of experiments. It can be expressed as

$$P(A) = \lim_{n \to \infty} \frac{m}{n},$$

where n is the number of times of the experiment is carried out and m is the number of times the event A happens.

2.1.2. Degrees of Belief and Subjective Probability

We often use "probability" to indicate the strength of our confidence or belief that the event will happen. A person who makes rational and consistent decisions, taking account of their believe and also the consequences of actions based on them, will in principle be able to associate numbers with their belief in a way which satisfies the axioms of probability. Such probabilities are described as "subjective" or "personal" [39,40,41].

These are all conditional probabilities, taking as given the information available to us at the time. If new evidence arrives which affects our assessment of the probability, our new assessment will again be conditional, this time taking as given the old information plus the new information. Different people, if they have different information, may therefore produce different estimates of the probability that a certain

event will happen. After sufficient trials, the subjective probability will agree with the objective probability However, the objective probability is not defined at all until we have carried out a large number of trials.

The conception of probability as a measure of personal belief is very important to research on the use of probability and decision theory for representing and reasoning with expert knowledge in expert systems. There is no alternative to acquiring from experts the bulk of probabilistic information used in expert system. Gathering a significant portion of frequencies through empirical study would entail much time and great expense. Nonetheless, probability theory provides for the gradual integration of appropriate statistical data into an expert system as it becomes available.

2.1.3. Bayes' Theorem

One of the most useful results of probability theory is Bayes' theorem. Bayes' theorem can be considered as a description of how probabilities are changed in the light of evidence. A common form of Bayes' theorem, which states that the probability of hypothesis H given evidence E is

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E \mid H)P(H)+P(E \mid \neg H)P(\neg H)},$$

where $P(H)$ is the prior probability of H and $P(H \mid E)$ is the posterior probability of H given the information E. Bayes' theorem establishes the connection between $P(H)$ and $P(H \mid E)$ and allows the adjustment of the probability of an event as evidence about it accumulates.

One of the attractive features of Bayes' rule is its amenability to recursive and incremental computation schemes

$$P(H \mid E, e) = P(H \mid E)\frac{P(e \mid E, H)}{P(e \mid E)},$$

where E denotes evidence observed in the past, and e denotes the new incoming evidence. The old probability $P(H \mid E)$ plays the role of the prior probability in the computation of new impact. The rule can be applied recursively if a chain of reasoning exists when hypothesis from one set of evidence is regarded as the evidence for a further hypothesis. This is call *Bayesian Inference*.

2.1.4. Application of Bayes' Theorem to Real Problems

Suppose that there are a set of hypotheses H=$\{H_1, H_2, ..., H_m\}$ and a set of evidence E=$\{E_1, E_2, ..., E_n\}$ in a problem. H could be, for example, possible diseases and E could be particular symptoms in a medical domain. What we are interested in is the posterior probability P(H | E), that is,

$$P(H \mid E) = P(H_1, H_2, ..., H_m \mid E_1, E_2, ..., E_n).$$

The probability $P(H \mid E)$ is an exponential function of both the number of hypotheses and the number of evidence. Bayes' theorem provides a mathematical way to calculate $P(H_{S_i} \mid E_1, E_2, ..., E_n)$

$$P(H_{S_i} \mid E_1, E_2, ..., E_n) = \frac{P(E_1, E_2, ..., E_n \mid H_{S_i}) P(H_{S_i})}{\sum_{H_{S_k} \subseteq H} P(E_1, E_2, ..., E_n \mid H_{S_k}) P(H_{S_k})},$$

where H_{S_i} is a subset of hypotheses H and H_{S_k} runs over all subsets of the power set of H. However, the probability $P(E_1, E_2, ..., E_n \mid H_{S_i})$ is also exponential both in the number of hypotheses and in the number of evidence, therefore, the complexity of the computation creates a problem in the application of Bayes' theorem.

In order to apply Bayes' theorem to real problems, two assumptions are often made. The first is that all evidence are conditionally independent, given any hypotheses. In other words, if the true hypothesis state was known precisely, then the likelihood of any evidence E_i would not depend on observations about any other evidence. Thus

$$P(E_i \mid H, E_1, ..., E_{i-1}, E_{i+1}, ..., E_n) = P(E_i \mid H).$$

It follows that

$$P(E_1, ..., E_n \mid H) = P(E_1 \mid H) P(E_2 \mid H, E_1) \cdots P(E_n \mid H, E_1, ..., E_{n-1}) = \prod_{i=1}^{n} P(E_i \mid H).$$

The second assumption is that hypotheses are mutually exclusive and exhaustive. More specifically, that one and only one hypothesis applies at any time. Instead of calculating $P(H \mid E_1, ..., E_n)$, we need only consider $P(H_i \mid E_1, ..., E_n)$, $H_i \in$ H.

With these two assumptions, the posterior probability of hypothesis can be simplified to be

$$P(H_i|E_1,....,E_n) = \frac{P(E_1|H_i)P(E_2|H_i)\cdots P(E_i|H_i)P(H_i)}{\sum_k P(E_1|H_k)P(E_2|H_k)\cdots P(E_i|H_k)P(H_k)}.$$

Only the conditional probabilities $P(E_i|H_k)$ and prior probability $P(H_k)$ are required for the computation, which is now tractable.

Prior probabilities are required in order to use Bayes' theorem. It may well be difficult to assess prior probabilities of hypotheses before evidence is available. Empirical data is often hard to obtain, and subjective judgement can be unreliable.

2.2. Bayesian Inference Systems

Bayes' theorem has been applied extensively to different domains such as medical diagnosis [4,7,38], mineral sites prospection [2,42], etc. We shall call these systems Bayesian systems. Basically, these systems can be divided into two groups: simple Bayesian system and proper Bayesian system. The only difference is that it is often assumed that some of the probabilities are independent in simple Bayesian systems.

Most of these Bayesian systems have shown success in some application domains. In particular, the simple Bayes method has been widely applied in a number of statistical systems for computer-aided diagnosis. Early in 1970's, de Dombal and his co-workers started their research on applying Bayes' theorem to the diagnoses of acute abdominal pain diseases [4,43,44,45]. The system was developed based on assumptions that the symptoms must be independent and later evaluated on a group of new patients with their correct final diagnoses. The system's decisions are compared with the actual correct diagnoses. Surprisingly, the accuracy of the system reached to 91.6%, which is far more accurate than human doctors' diagnoses [45].

The GLADYS (GLAsgow DYspepsia System) is a system developed by Spiegelhalter and Knill-Jones [7] for diagnosing patients with dyspepsia problems. Similar to the above system, it uses the simple Bayes's formula to make inference. However, instead of using conditional probabilities to update one's belief in a certain disease, they assigned to each symptom one or more "weights of evidence" for or against a disease. When a certain symptom is observed in a patient, the weight of this symptom for or against a

certain disease can be calculated and will be added to the weight of the corresponding disease. The higher the weight of evidence is for a certain disease, the more likely it will be present in the current patient.

PATHFINDER is an expert system to assist general pathologists with diagnosis in the area of hematopathology [46]. Early work on PATHFINDER explored a variety of nonprobabilistic and quasiprobabilistic schemes [47]. Finally, PATHFINDER investigators tried a Bayesian probabilistic scheme, and built on the principles of probability and decision theory. The current PATHFINDER includes about 60 diseases of the lymph node and over 130 features that can be observed to make a diagnosis [46]. The model makes use of 75,000 probabilities in performing inference. The use of similarity networks and partitions have made it practical to encode this amount of expert knowledge without unreasonable effort [48].

In the rest of the section, we are going to review two Bayesian systems, the PROSPECTOR model and the G&T model, to see how they applied Bayes' theorem to manage uncertainty and how they dealt with the computational complexity problem.

2.2.1. PROSPECTOR Model

PROSPECTOR is an expert system which helps geologists evaluate the mineral potential of exploration sites [2,42]. Rules are used for representing the domain knowledge and Bayes' theorem, in odds form, for evidence propagation.

The system rule format is:

IF E THEN H (to degree C),

where E is an arbitrary logical expression and H is a hypothesis. This rule is interpreted to mean "the observed evidence E suggests the hypothesis H to the degree C". "C" establishes the "strength" of the rule and specifies how the probability of H is to be updated given that of E. All the rules are then represented by an inference network. Nodes in the inference network are either evidence E or hypothesis H. To reason with a rule "IF $E_1, E_2, ..., E_n$ THEN H", PROSPECTOR first combines all the E_i into a single evidence E and then performs the deductive inference for the rule "IF E THEN H".

Given a hypothesis H and evidence E, we have according to Bayes' rule

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E \mid H)P(H)+P(E \mid \neg H)P(\neg H)}$$

and we get the odds-likelihood formulation of Bayes' rule:

$$O(H \mid E) = LS(E \mid H)\, O(H),$$

$$O(H \mid \neg E) = LN(E \mid H)\, O(H),$$

where

O(H) is the prior odds on the hypothesis H,

O(H|E) and O(H|¬E) are the posterior odds on the hypothesis H (given that the evidence E is observed to be present and absent, respectively).

LS and LN are sufficiency factor and necessity factor respectively [2,42]. They are defined by:

$$LS(E \mid H) = \frac{P(E \mid H)}{P(E \mid \neg H)},$$

$$LN(E \mid H) = \frac{P(\neg E \mid H)}{P(\neg E \mid \neg H)}.$$

The sufficiency factor is the degree to which learning that E is true is sufficient for believing that H is true, and the necessity factor is the extent to which learning that E is true is necessary for believing H. The experts are asked to estimate these two measures for each rule.

Assume that there is a collection of evidence, E_1, E_2, ..., E_n. Let H stand for the hypothesis of interest. The sufficiency factor is calculated from the following formulae:

$$LS(E_1, E_2, ..., E_n \mid H) = \frac{P(E_1, E_2, ..., E_n \mid H)}{P(E_1, E_2, ..., E_n \mid \neg H)}.$$

After all the evidence has been examined, the combined belief in the hypothesis H is calculated by

$$O(H \mid E_1, E_2, ..., E_n) = LS(E_1, E_2, ..., E_n \mid H)\, O(H).$$

We need 2^n-1 probability values to specify fully the probabilities of combination for every subset of evidence conditioned on H, that is, $P(E_1, E_2, ..., E_n \mid H)$. In general, it is very difficult to obtain all the relevant probabilities in real application areas. Moreover it is not computationally feasible to manipulate

these probabilities. In order to reduce these difficulties, the independence assumption (discussed in section 2.4) is made in the system. Therefore we can write

$$P(E_1, E_2, ..., E_n \mid H) = \prod_{i=1}^{n} P(E_i \mid H)$$

and

$$P(E_1, E_2, ..., E_n \mid \neg H) = \prod_{i=1}^{n} P(E_i \mid \neg H)$$

which lead to

$$O(H \mid E_1, E_2, ..., E_n) = O(H) \prod_{i=1}^{n} LS(E_i \mid H) .$$

So instead of 2^n-1 probability values, the number of probabilities decreases to n, just one for each E_i.

Under the conditional independence assumption, we have

$$P(e \mid E, H) = P(e \mid H) \text{ and } P(e \mid E, \neg H) = P(e \mid \neg H) .$$

A simple recursive procedure for updating the posterior odds is

$$O(H \mid E, e) = O(H \mid E) LS(e \mid H) .$$

In general, the evidence E is often not known with certainty. For example, the user is often unable to observe either the definite presence and absence of the evidence in practical situations. Typically, the user is prepared only to indicate a degree of belief that the evidence being encountered is actually present. In this case, Bayes' theorem can not be applied directly to this problem. Any systems that employ Bayesian inference as their inference mechanisms would have to deal with this problem. For example, PROSPECTOR uses a piecewise linear interpolation between the two extreme cases of perfect certainty as discussed below [12].

Let E' denote the observations that cause the user to suspect the presence of the evidence E and $P(E \mid E')$ stand for the uncertain observation. We have

$$P(H \mid E') = P(H, E \mid E') + P(H, \neg E \mid E') = P(H \mid E, E') P(E \mid E') + P(H \mid \neg E, E') P(\neg E \mid E') .$$

It seems reasonable to presume that, if we knew whether or not E was true, knowledge about the observation E' would provide no additional information to H. That is,

$$P(H \mid E') = P(H \mid E)P(E \mid E') + P(H \mid \neg E)P(\neg E \mid E'), \tag{2.1}$$

where $P(H \mid E)$ and $P(H \mid \neg E)$ are obtained directly from odds-likelihood formulation of Bayes' rule.

However, this interpolation scheme (2.1) between two values $P(H \mid E)$ and $P(H \mid \neg E)$ can not be directly adopted since the computation is based on the subjective probabilities provided by domain experts [42]. The resulting value of $P(H \mid E')$ will usually not agree with all expert's estimate for the prior probability P(H). Therefore, an ad hoc function, a piecewise linear interpolation function of $P(E \mid E')$ is used in PROSPECTOR so that the desired values are obtained at the three fixed points $P(E \mid E')$=0, P(E) and 1. The resulting function is depicted in Figure 2.1. The differences between the broken and unbroken lines illustrate the discrepancies between a formal and a subjective Bayesian updating.

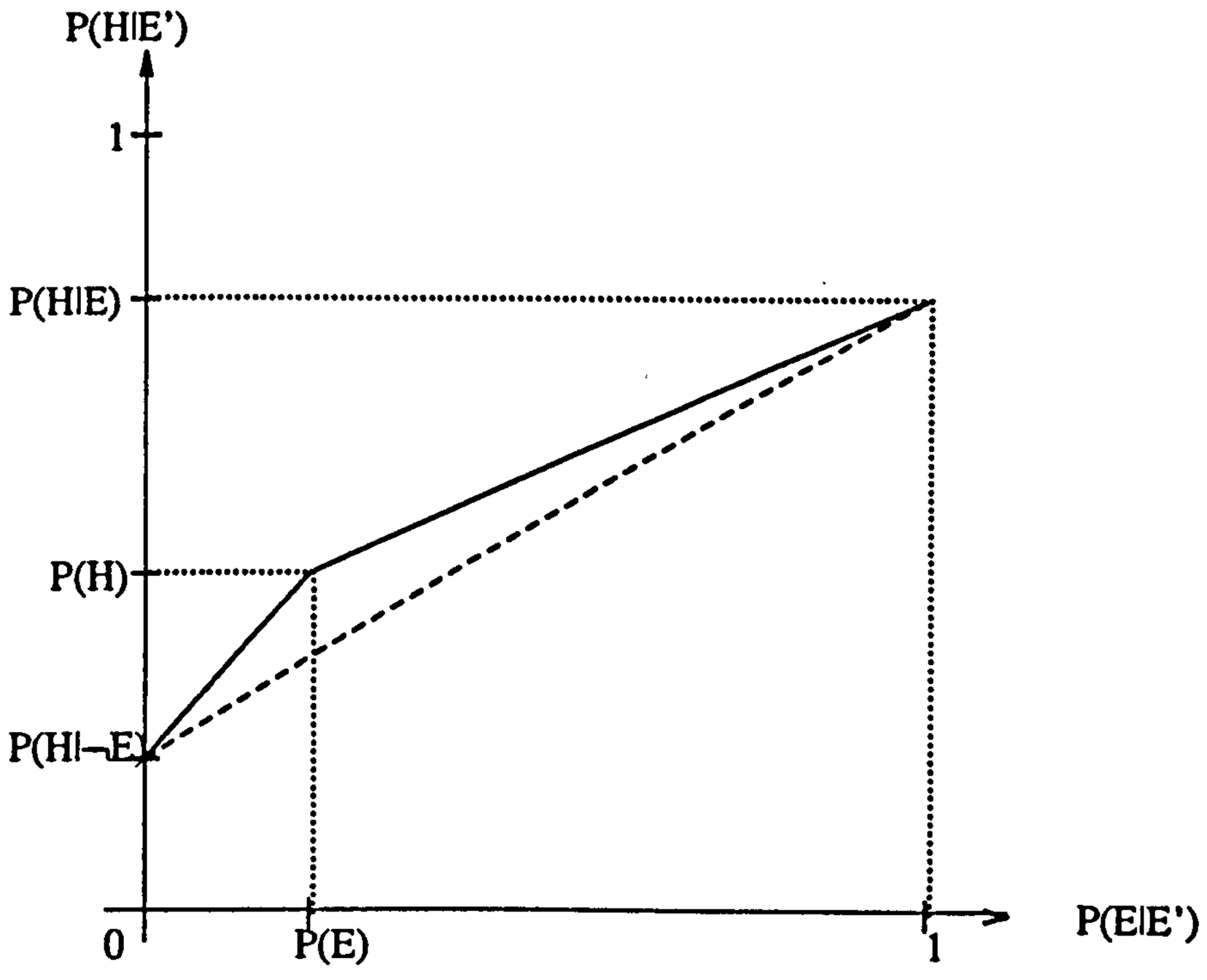


Figure 2.1. Consequent Probability as a Function of the Antecedent Probability

Although Bayesian inference has been successfully employed for use in PROSPECTOR, unfortunately, real world problems often do not conform to the assumption that items of evidence are conditionally independent to hypothesis. It seems unreasonable to make such an assumption in the context of expert systems. Moreover, it is very difficult for domain experts to estimate prior probabilities. Ad hoc function adjustments are used in the system [12] to treat inconsistency. Presently, the exact consequences of using this ad hoc function in application areas is not known.

2.2.2. G&T Model

Attempts to overcome some of the difficulties arising from Bayesian inference have been made. Variant approaches have been proposed [7,14,15,16,49,50]. If the past data is available and sufficiently rich in content and quantity, it is plausible to cope with uncertainty without assuming independence. G&T or "Proper Bayes" [38,51] is one such model. This method has been successfully applied to a large medical database consisted of abdominal pain patients [22,38].

The idea is essentially to analyse the past data, and select the most relevant combinations for each group in this data by means of statistical methods. When a new case arrives, significant features are detected and their probabilities of belonging to each of the possible groups described by the past data evaluated. Classification results can be achieved, for example by selecting the group with the highest probability amongst the possible groups [38].

Consider, for example, the problem of estimating from the past data the probabilities that patients have certain diseases, given their symptoms. In general, if D_i is ith possible disease and S is the combination of symptoms we observed, we have:

$$P(D_i|S) = \frac{P(D_i, S)}{\sum_i P(D_i, S)} = \frac{No.\ of\ patients\ with\ D_i\ and\ S}{No.\ of\ patients\ with\ S}. \qquad (2.2)$$

We can work out $P(D_i, S)$ from the past data. So an estimate of the probability that a new patient with symptoms S will have the disease D_i is calculated.

Because of the limited number of past patient examples, we can not expect to find enough cases with exactly the same symptoms as those observed for a new patient. In addition, not every symptom of the new patient is crucial for diagnosing the disease. The important thing is whether this patient has significant symptoms of some diseases and what are these possible symptoms. Statistical methods are used to select the most relevant symptoms for each disease in the database. For example, the selection of relevant symptoms could be based on a chi-square (χ^2) measure. The most significant symptom for a disease is the one that has the maximum chi-squared value, positive or negative. Suppose that symptom S_i is chosen as the most relevant symptom to the disease D. The patients with symptom S_i and without S_i are then treated differently. In other words, there are two groups now: group 1 consisted of patients with S_i and group 2

containing those without S_i. The second relevant symptom to the disease D is found for each of the two groups respectively. The selection process is carried on until the chi-squared value of all the remaining symptoms are lower than a threshold given by the user. The selection process is best represented by the following diagram:

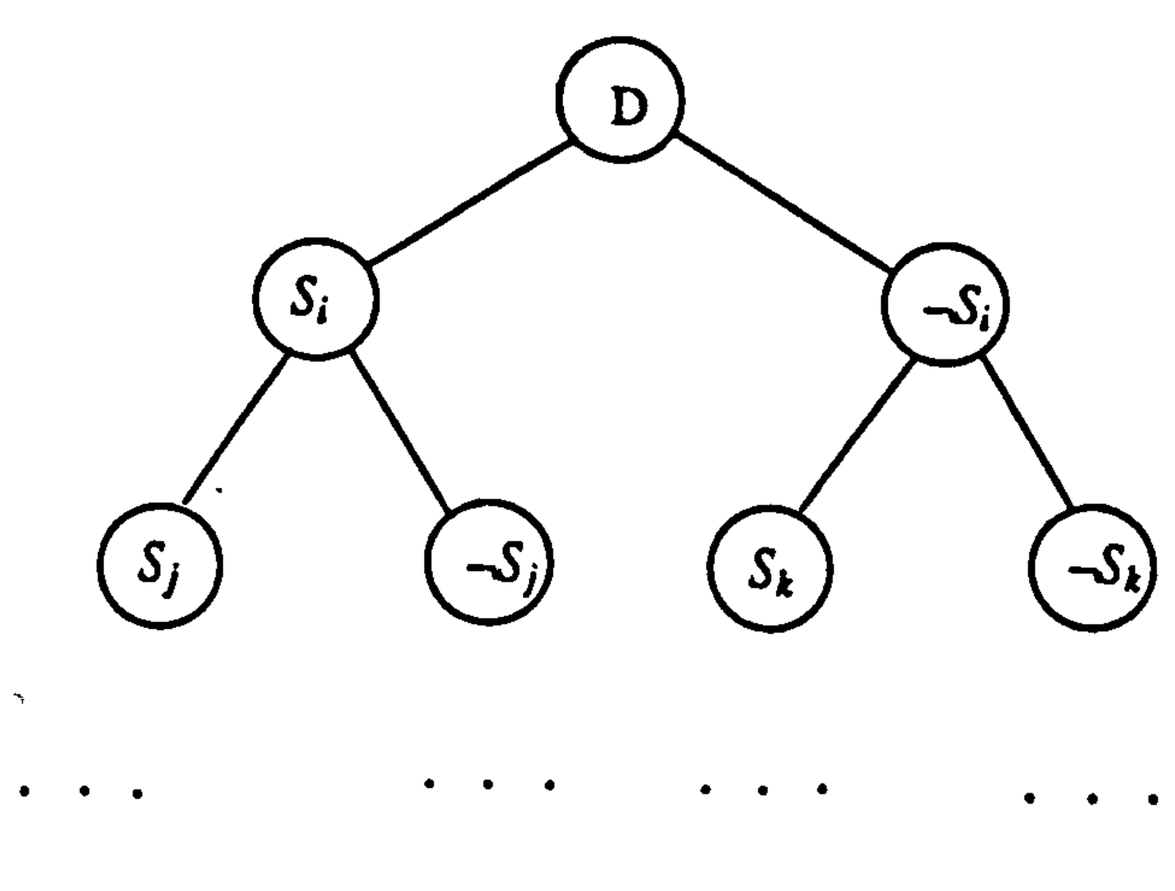


Figure 2.2. The G&T Selection Process

From the above diagram, possible relevant symptom combinations are extracted, namely all the possible paths from the root (D) to each leaf. Probabilities are calculated for the obtained combinations. These probabilities are estimated by the frequencies in the database according to the formula (2.2). The precision of such estimation is measured by upper and lower bound confident limits [52]. The knowledge is extracted and expressed in a rule form: IF the combination of symptoms is present, THEN the patient will have disease D with probability p.

When a new patient comes, we shall normally know whether he/she has or does not have each of possible symptoms. These symptoms are then compared with those combinations extracted for each disease. The probabilities associated with those matched combinations can stand for his/her likelihood of having the corresponding disease. The higher the probability a certain disagnostic group has, the more probable it is that the patient would be diagnosed with this disease. Therefore, the diagnostic group with the highest probability is selected as the computer diagnosis for this new patient.

The G&T method is extremely straightforward and easily explained. The probabilities used are derived directly from the database using Bayes' theorem without assuming any independence. So experts are not required to make any subjective estimates. Unfortunately this approach is only applicable to

domains where a large amount of accumulated data is available. The performance of the system is dependent on a particular population. Furthermore the system does not presently handle uncertain evidence.

Although the G&T method was developed completely independently, it was subsequently found to have some marked resemblances to some inductive machine learning algorithms. Classification trees are used to make decisions in these methods. Quinlan's ID3 algorithm is one of the earliest proposed methods in the field of inductive learning [53,54]. CART [55], a system built based on *Classification And Regression Tree*, is another example of the inductive learning methods.

2.3. Graphical Knowledge Representations

Recent work in theoretical statistics has shown that it is possible to adopt a sound probabilistic approach to uncertain reasoning using graphical knowledge representations [26,28]. In contrast to the Bayesian inference models discussed above, the knowledge about a domain is expressed in a graphical form, namely Bayesian belief networks [25,28,56]. Qualitative dependent relationships are expressed explicitly in the graph and quantified by conditional probabilities. Such a representation has many virtues due to the transparency of the knowledge embedded and its ability to unify almost all domain knowledge relevant for an expert system. In this section, Bayesian belief networks are introduced and related graphical knowledge representations are discussed. The basic notations in graph theory to understand Bayesian belief networks are presented in Appendix A. The role of Bayesian belief networks as a representation of conditional independence is also investigated in Appendix A.

2.3.1. Bayesian Belief Networks

Bayesian belief networks have been studied intensively recently and named differently in the field of artificial intelligence: Bayes networks [25], belief networks [57], causal graph [26], causal probabilistic network [58], causal networks [28], probabilistic causal network [59], recursive causal networks [60], and so on.

Formally, a Bayesian belief network can be represented as a pair <G, P>. Qualitative knowledge G is a directed acyclic graph (DAG) <V, R> constructed from a probability distribution P, where V is a set of nodes,

$$V = \left\{ V_1, V_2, ..., V_n \right\}.$$

The nodes V of the graph G are random variables in the domain. Each of the variables may take a finite set of values. R is a set of dependency relationships among the variables. For each node V_i, if V_j is a direct influence of V_i, there is a directed link $V_j \rightarrow V_i$. This kind of relationship is specified in R

$$R = \left\{ V_j \rightarrow V_i \quad | \; V_i, V_j \in V \;\text{ and }\; V_j \textit{ is a direct influence on } V_i \right\}.$$

This shows that V_j is a parent node of V_i. The topmost (or root) nodes (i.e. without any links pointing to them) have an empty set of influential nodes. Absent links indicate conditional independence assumptions reflected in the network. An example of a Bayesian belief network is depicted in Figure 2.3.

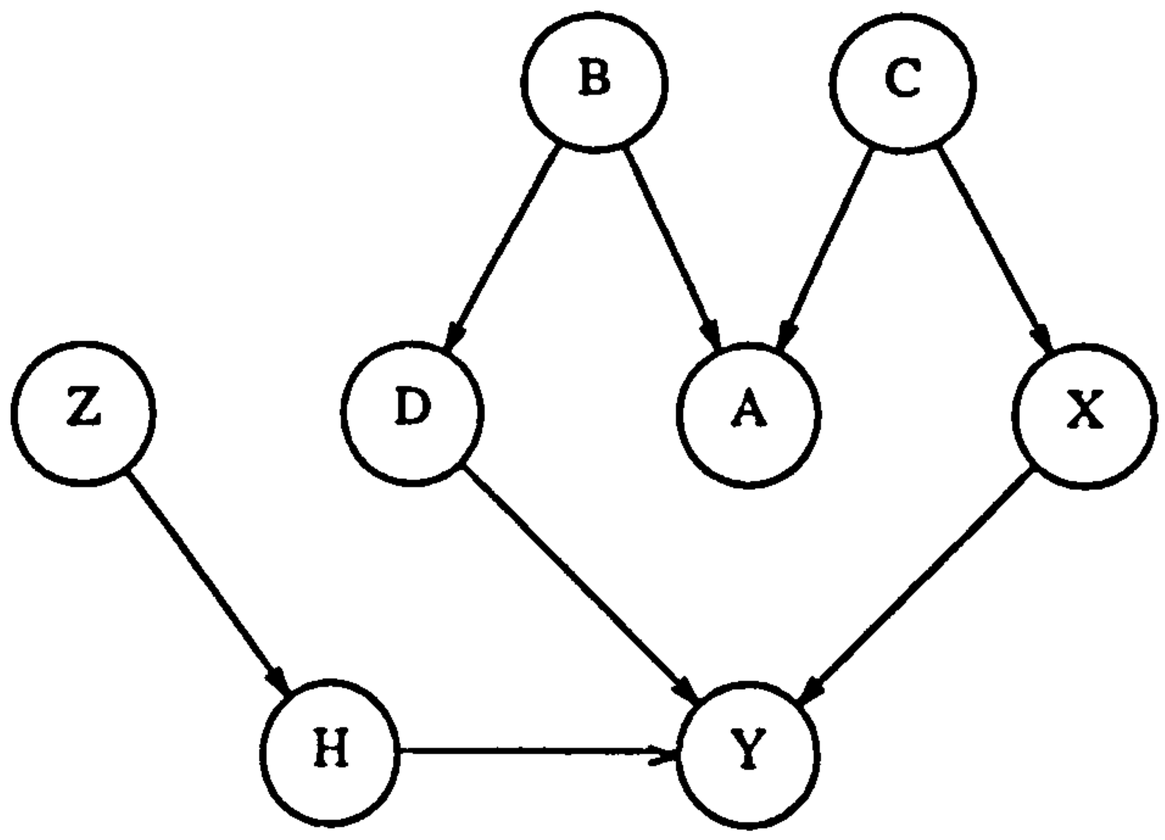


Figure 2.3. An Example - Bayesian Belief Network

Quantitative knowledge P is a set of conditional probability distributions on G. Each node in G corresponds to a variable in P and a set of nodes correspond to a set of variables. Each node V_i in the graph may have a set of parent nodes $PA(V_i)$ and has a conditional probability table. So the conditional probability table (CPT) has the following form:

$$CPT = \left\{ P(V_i \mid PA(V_i)) \mid V_i \in V \text{ and } PA(V_i) \textit{ is a set of parents of } V_i \textit{ in } G \right\}.$$

If V_i has no parents ($PA(V_i) = \varnothing$), then $P(V_i \mid PA(V_i)) = P(V_i)$. CPT must satisfy two conditions

1) $P(V_i \mid PA(V_i)) \geq 0$;

2) $\sum_{V_i} P(V_i \mid PA(V_i)) = 1$.

Bayesian belief networks capture both qualitative and quantitative knowledge. At the qualitative level, a Bayesian belief network has built-in independence assumptions. In general, an edge denotes a probablistic dependence between linked variables. More precisely, the absence of a directed edge denotes various kinds of conditional independence between variables. These independence assumptions determine what quantitative information is required to specify the probability distribution P among the random variables in the network G.

At the quantitative level, P consists of a set of conditional probability tables. Each node has a table to store a number of conditional probabilities, which fully specify the probability of this node given all the combinations of its parents. Two sorts of probabilities are held: (i) A variable which is not dependent on the values taken by other variables requires probabilities associated with all the possible values it may take (However, since these probabilities have to sum to 1, once all except one are known the last follows automatically); (ii) A variable which depends on the values taken by other variables requires probabilities associated not simply for all possible values it may take but for all possible values which the variables on which it depends may take. For example, if we have a Bayesian belief network depicted in Figure 2.3, the following probabilities need to be specified: (i) P(B), P(C), P(Z); (ii) P(D|B), P(A|B,C), P(X|C), P(H|Z), P(Y|D,X,H).

Therefore, it is a nice property of Bayesian belief networks that if you specify the required conditional probabilities consistently, the joint probability distribution will be consistent and the network will define uniquely a distribution. It is not too hard to see that the claim is true as the underlying distribution from which a Bayesian belief network is constructed can be expressed as [25,28]

$$P(V_1, V_2, ..., V_n) = \prod_{i=1}^{n} P(V_i \mid PA(V_i)) .$$

In this way, the Bayesian belief network provides a simple solution to the unsolved problem in the PROSPECTOR model. These conditional probability tables form the fundamental components from which case-specific inferences are eventually derived, and are the natural parametrisation for initialisation from domain expert opinions or available data.

2.3.2. Related Graphical Knowledge Representations

An influence diagram (ID), which is essentially a Bayesian belief network, has the addition of decision nodes, deterministic nodes and value nodes [24,27,61]. A deterministic node represents a state of the world that is a deterministic function of its predecessor nodes. The decision nodes correspond to the set of actions available to the decision maker and the value node represents the objective to be maximised in expectation. Generally, each influence diagram has only one value node. Its predecessors indicate those outcomes or attributes that are included in the evaluation of a choice or plan. The relationships among random variable nodes (chance nodes), decision nodes and the value node are represented explicitly in the ID. Influence diagrams are developed as a computer-aided modeling tool and used to support the calculation of the decisions or strategies that maximise expected value (or more generally utility). The modern development of influence diagrams is largely due to the work of Miller et al [61], Olmsted [62], and Howard and Matheson [24].

Since it may be difficult to obtain a complete and precise point-valued probability distribution, some general representations of dependence relationships among variables have been explored. A purely qualitative representation, qualitative probabilistic network(QPN), is one of the examples [63,64]. QPN is an abstraction of Bayesian belief networks and specifies the signs of monotonic and synergistic probabilistic influences among variables. Therefore it provides a basis for qualitative probabilistic inferences about the directions of effects on belief and decisions of evidence.

Fertig and Breese [65] introduced a generalisation of point-valued probabilities using a particular class of interval-valued probability distributions. Lower-bounds are specified for each conditional probability. The upper bound on each probability is implicit in the lower bounds.

DAGs have also been used as a tool for representing Dempster-Shafer belief functions to facilitate inference where the available evidence does not support a full probabilistic specification [66,67,68,69].

2.4. Reasoning Algorithms in Bayesian Belief Networks

It is necessary to exploit and take full advantage of the independence relationships embodied in the network, and find efficient evidence propagation algorithms. The basic idea is to decompose the global computation of the joint probability distribution into a series of computations on small groups [25,26,28,70]. The computations on these groups, referred to as "local computations", work on, at any one time, a variable and its neighbours in the graphical structure. These computations can be carried out step by step and the joint probability distribution can be easily obtained from these local computations. Propagation is achieved by having these small groups which can send messages to each other and perform necessary operations as a result of received messages.

A variety of reasoning methods have been developed, each focusing on particular families of Bayesian belief network topology [25,26,28,31,32,71]. In this section, three evidence propagation algorithms using Bayesian belief networks are studied. Computational algorithms are developed and investigated.

Thus, this section is arranged as follows. In section 2.4.1, Pearl's algorithm is presented. In section 2.4.2, Lauritzen and Spiegelhalter (L-S) algorithm based on a reformulation of Bayesian belief networks is discussed. In section 2.4.3, a comparison between Pearl's algorithm and the L-S algorithm is made. In section 2.4.4, an approximate algorithm - stochastic simulation - is reviewed. Other related algorithms are discussed in section 2.4.5.

2.4.1. Pearl's Message Passing Algorithm

There is one and only one undirected path between any two nodes in a singly connected structure (polytree). A node V_j blockades a path between its parent nodes and child nodes. In other words, V_j's parent nodes are conditional independent of V_j's child nodes given V_j. This property is very useful in calculating posterior probabilities. The posterior probability of V_j given evidence E can be represented as

$P(V_j|E)$, which is known as $Bel(V_j)$ as well, where evidence E is a set of instantiated nodes. Consequently, E can be divided into two parts: E^-_{vj} and E^+_{vj}, which represent the subset of evidence E in the tree rooted at V_j and the remainder of the tree respectively. That is, E can be expressed as $E^+_{vj} \cup E^-_{vj}$, where the symbol $\cup$ represents a union of variable sets. According to Bayes' rule, we have

$$P(V_j|E) = P(V_j|E^+_{vj}, E^-_{vj}) = \frac{P(V_j, E^+_{vj}, E^-_{vj})}{P(E^+_{vj}, E^-_{vj})}$$

$$= \frac{P(E^+_{vj})P(V_j|E^+_{vj})P(E^-_{vj}|V_j, E^+_{vj})}{P(E^+_{vj}, E^-_{vj})}.$$

By conditional independence assumption in the structure, $P(E^-_{vj}|V_j, E^+_{vj}) = P(E^-_{vj}|V_j)$. Therefore we have

$$P(V_j|E) = \frac{P(E^+_{vj})}{P(E^+_{vj}, E^-_{vj})} P(V_j|E^+_{vj})P(E^-_{vj}|V_j),$$

where $\frac{P(E^+_{vj})}{P(E^+_{vj}, E^-_{vj})}$ is a normalising constant and can be denoted by α. If we define two messages for variable V_j: $\pi(V_j)=P(V_j|E^+_{vj})$ and $\lambda(V_j)=P(E^-_{vj}|V_j)$, we get

$$P(V_j|E) = Bel(V_j) = \alpha\pi(V_j)\lambda(V_j). \quad (2.3)$$

That means we can calculate the posterior probability of V_j given evidence E by only using two messages π and λ, where $\pi(V_j)$ represents the causal supports from the parent nodes and $\lambda(V_j)$ represents the diagnostic supports from V_j's child nodes [25]. The computation of the posterior probability is local in the sense that only its parent and child nodes are involved.

We have the following conclusions following the definition of the two messages. Initially, for any topmost (root) node V_R, $E^+_{V_R}$ to be empty, $\pi(V_R)=P(V_R|\varnothing)=P(V_R)$. On the other hand, for a leaf node V_L, $E^-_{V_L}$ to be empty, $\lambda(V_L)=P(\varnothing|V_L)=1$.

Based on these observations, Pearl's algorithm is developed [25,70]. For simplicity, we assume that V_j has two parents F_1, F_2 and two children C_1 and C_2, see Figure 2.4. It is clear that E^-_{vj} can be partitioned into E^-_{C1} and E^-_{C2}, and that E^-_{C1} is conditionally independent of E^-_{C2} given V_j.

The message $\lambda(V_j)$ defined as $P(E^-_{vj}|V_j)$ can be written as

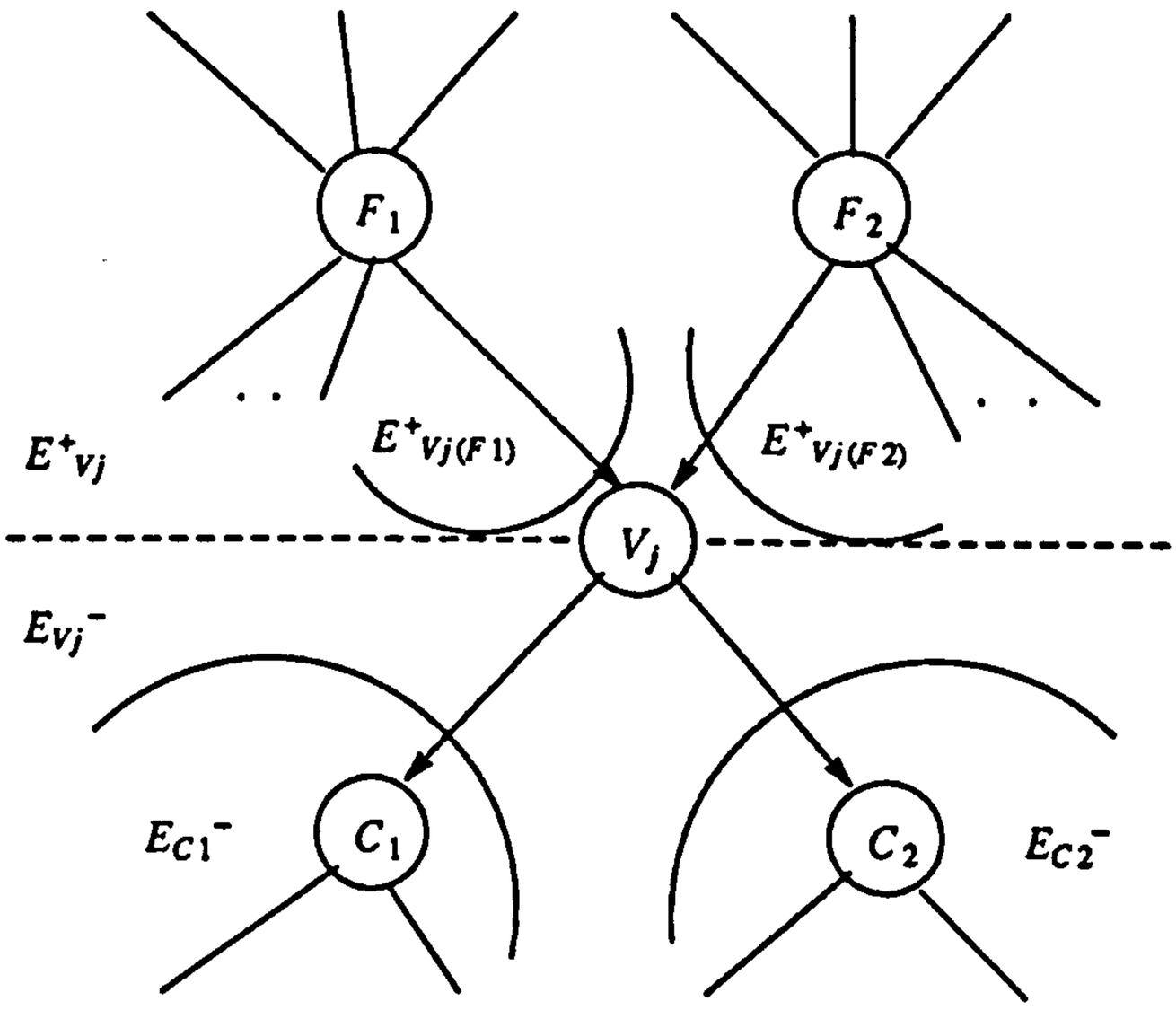


Figure 2.4. A Singly Connected Structure

$$\lambda(V_j) = P(E_{C1}^- \cup E_{C2}^- | V_j) = P(E_{C1}^- | V_j) P(E_{C2}^- | V_j, E_{C1}^-)$$

$$= P(E_{C1}^- | V_j) P(E_{C2}^- | V_j) .$$

Once again, we define:

$$\lambda_{Ci}(V_j) = P(E_{Ci}^- | V_j) \qquad (i=1, 2) ,$$

as a message that a child C_i sends to its parent V_j. Therefore, we get

$$\lambda(V_j) = \prod_{i=1}^{2} \lambda_{Ci}(V_j) . \tag{2.4}$$

Now considering how to calculate $P(E_{Ci}^- | V_j)$, we have:

$$P(E_{Ci}^- | V_j) = \sum_{Ci} P(E_{Ci}^- | V_j, C_i) P(C_i | V_j) \qquad (i=1, 2) .$$

By the conditional independence embedded in the structure, $\lambda_{C1}(V_j)$ for example is calculated

$$\lambda_{C1}(V_j) = P(E_{C1}^- | V_j) = \sum_{C1} P(E_{C1}^- | C_1) P(C_1 | V_j) = \sum_{C1} \lambda(C_1) P(C_1 | V_j) . \tag{2.5}$$

So message $\lambda_{C1}(V_j)$ can be calculated by C_1's λ message and the conditional probability of C_1 given V_j.

Again, message $\pi(V_j)$ can be calculated from its parents F_1 and F_2.

$$\pi(V_j) = P(V_j \mid E^+_{V_j}) = \sum_{F_1, F_2} P(V_j \mid F_1, F_2, E^+_{V_j}) P(F_1, F_2 \mid E^+_{V_j})$$

$$= \sum_{F_1, F_2} P(V_j \mid F_1, F_2) P(F_1, F_2 \mid E^+_{V_j(F_1)}, E^+_{V_j(F_2)})$$

$$= \sum_{F_1, F_2} P(V_j \mid F_1, F_2) P(F_1 \mid E^+_{V_j(F_1)}) P(F_2 \mid E^+_{V_j(F_2)}) .$$

By definition

$$\pi_{F_i}(V_j) = P(F_i \mid E^+_{V_j(F_i)}) \qquad (i=1, 2),$$

we have

$$\pi(V_j) = \sum_{F_1, F_2} P(V_j \mid F_1, F_2)\, \pi_{F_1}(V_j) \pi_{F_2}(V_j) , \tag{2.6}$$

where $\pi_{F_1}(V_j)$ is a message sent by V_j's parent F_1 and can be calculated by

$$\pi_{F_1}(V_j) = P(F_1 \mid E^+_{V_j(F_1)}) = P(F_1 \mid E^+_{F_1}) P(E^-_{F_1} - E^-_{F_1(V_j)} \mid F_1) ,$$

where the symbol – denotes a subtraction of variable sets.

$$\pi_{F_1}(V_j) = \alpha \pi(F_1) \prod_{Vi \neq Vj} \lambda_{Vi}(F_1) \qquad \textit{where } V_i \textit{ is a child of } F_1. \tag{2.7}$$

It has been shown that the method has the desirable property of calculating the updated probabilities only through communication between neighbouring nodes and using message passing techniques. Initially, message λ is set to be a unit vector for each node. Only message π is needed to be passed around the polytree. Therefore, starting from the top most nodes, the initial marginal probability of each node can be obtained using the method described.

2.4.1.1. Evidence Propagation

Having assimilated the required information, the algorithm will be ready to propagate evidence. We interpret evidence as information or observation for a particular variable in Bayesian belief networks. The evidence can be either certain or uncertain. Uncertain evidence is also called virtual evidence [25] or uncertain observation [28]. Certain evidence is the observation that we are 100% sure of what has been obtained. When certain evidence arrives, the messages on observed nodes will have been changed. For example, if evidence variable E is observed as E_j, the following procedure is carried out to absorb evidence

BEGIN
set $Bel(E_j)=\lambda(E_j)=\pi(E_j)=1$ and for i≠j set $Bel(E_i)=\lambda(E_i)=\pi(E_i)=0$
send a new λ message to E's parent by expression (2.5)
send a new π message to E's child by expression (2.7)
END

The impact will then be propagated. The propagation algorithm is as follows:

BEGIN
WHILE not all nodes are updated DO
BEGIN
IF a variable B receives a new λ message from one of its children
AND B is not already observed
THEN
BEGIN
compute the new value of λ(B) by expression (2.4)
compute the new value of *Bel* (B) by expression (2.3)
send a new λ message to B's parents by expression (2.5)
send a new π message to B's other children by expression (2.7)
END
IF a variable B receives a new π message from one of its parents
AND B is not already observed
THEN
BEGIN
compute the new value of π(B) by expression (2.6)
compute the new value of *Bel* (B) by expression (2.3)
send a new λ message to B's other parents by expression (2.5)
send a new π message to B's children by expression (2.7)
END
END
END

Often people are fairly certain that evidence E is observed as false, but they still wish to see the influence of a small chance of true. In this case, the probability is used to represent the level of uncertainty of observation [25,28].

The uncertain evidence E' on V_j can be represented as the relative magnitudes of the terms $P(E'|V_j)$ [25]. Since the absolute magnitudes do not affect the calculations, we can update the beliefs as though this likelihood vector originated from an ordinary, logically definable event E. The estimate $P(E'|V_j)$ will be treated as a λ message sent to V_j [25].

BEGIN
calculate a new λ message for V_j
calculate a new belief $Bel(V_j)$
send a new λ message to V_j's parent by expression (2.5)
send a new π message to V_j's child by expression (2.7)
END

The evidence propagation algorithm can then be applied.

2.4.1.2. Multiply Connected Networks

The algorithm described works only for a singly connected structure (polytree). However, this is not the case in many problems. When the networks are multiply connected possible cycling of information is a problem when using Pearl's algorithm directly.

There are some ways to apply this algorithm to multiply connected networks [29]. The basic idea is to change the underlying network into that of the required structure and then to apply Pearl's algorithm. One possible solution, called "conditioning", is to find a set of loop cut nodes [29,72]. These nodes are then assumed to be observed, and therefore, they can be removed leaving the resultant network singly connected. In other words, we decompose the multiply connected network into a number of singly connected networks. The number of possible instantiations becomes the number of possible combinations of values that the members of the loop cut sets can take. When we observe a new piece of evidence, the information in each network must be updated independently. The results of these independent calculations are then combined by the conditional probability of the nodes in the loop cut set C, given evidence E. That is:

$$P(V_i \mid E) = \sum_C P(V_i \mid E, C) P(C \mid E),$$

where $P(V_i \mid E, C)$ is obtained by Pearl's algorithm for singly connected networks and $P(C \mid E)$ can be derived from the prior probability $P(C)$ as follows [29]

$$P(C \mid E) = \alpha P(E \mid C) P(C),$$

where α is a normalising constant and $P(C)$ and $P(E \mid C)$ can be calculated using the initialisation method described in [73]. It is clear from the above expression that the time complexity of this method is exponential in the number of nodes in the loop cut set. An upper bound on the time complexity of the initialisation is $O(LMN^2)$, where L is the product of the number of possible values of every node in the loop cut set, M is the maximal product of the number of possible values of any node and the number of possible values of each of its parents, and N is the number of nodes in the network. It is necessary to find a small loop cut set for the multiply connected network to minimise the value of L.

There is a heuristic method that attempts to find a possible small loop cut set [72]. It contains the following main steps:

1. delete any links that are not in any loop

2. if there are any nodes left, find a loop cut node

    2.1. add the node to the loop cut set and remove it from the network

    2.2. go to 1

3. terminate when no nodes remain in the network

The criteria of finding a loop cut node has three aspects: (i) the node has one or fewer parent nodes; (ii) the node has the most neighbours; (iii) the node has the least set of values. We attempt to minimise the number of instantiations of the loop cut set. However, the method is not guaranteed to be minimal. The worst time complexity for finding a loop cut set using this algorithm is $O(N^2)$.

2.4.2. Lauritzen-Spiegelhalter's Clique Algorithm

Pearl showed how if the structure is a polytree, then the required results could be obtained using only "local computations" without calculating the full joint distribution. The algorithm computes a posterior marginal distribution for each variable by visiting each node at most once for each piece of evidence. Furthermore, there is no need to have an "overall controller" of the process; each node communicates autonomously with its neighbours in the structure.

Lauritzen and Spiegelhalter [28] deal with the problem strictly as a mathematical one and show that local computations can still be used in multiply connected networks. They explore the theory of Markov fields [74,75] in which probabilistic conditional independencies are related to undirected graphs [76]. This theory rests on the relationship between the joint probability distribution form and the cliques of the triangulated graph. They emphasise systematic re-representation of Bayesian belief networks and conditional probability tables (secondary structure) for computational purposes.

2.4.2.1. Initialisation

The re-representation of graphical structure is based on Markov field theory and carried out by a set of truth-preserving graphical manipulations. It means that the joint probability distribution represented by the revised model is implied by the structure in the original probabilistic model. In this case, making the original graph triangulated is the most important operation of graphical manipulations [28]. The methods of making graphs triangulated discussed in Appendix A are used.

The restructuring of the network can be achieved by a series of graphical manipulations: marrying parents, dropping direction and triangulating the graph, resulting in an undirected triangulated graph being extracted from the Bayesian belief network. Then the creation of a clique tree starts with identifying cliques (C), obtaining a clique ordering with the running intersection property. Next, separators (S), residuals (R) and possible parent cliques for each clique are found. Finally, a clique tree is constructed where nodes represent cliques and undirected edges represent separators. The attractive consequence of constructing a clique tree is that all necessary computations and storage for the joint probability distribution can be carried out in a coherent probabilistic manner using local computations on the clique tree [30,31].

Once the clique tree is built, it is stored until the Bayesian belief network is modified. The tree is therefore a permanent part of the system and serves as the computational data structure. All the probabilities are determined from the tree rather than from the Bayesian belief network. The cliques are the objects and the separators are the communication channels between them.

Having changed the graphical structure (qualitative representation), the corresponding numerical (quantitative) representation changes. We have seen how the joint probability distribution was originally expressed as a product of conditional probabilities of nodes given their parents, that is,

$$P(V) = \prod_{V_i} P(V_i \mid PA(V_i)) . \tag{2.8}$$

The algorithm emphasises an initial restructuring of the network into a form that connects explicitly the set of variables, on which functions will have to be calculated as intermediate steps in obtaining any desired conclusion, in a new undirected representation. Such general functions are called "potential functions", which take non-negative values. A potential function is defined as a function ψ (or ϕ) mapping space on a

finite set of variables V onto the unit interval [0, 1]. The ψ functions are nonnormalised assessments of joint probabilities,

$$P(V) = \frac{1}{Z}\psi(V),$$

where $Z=\sum_{V}\psi(V)$. Now we need to express the joint distribution as a product of potential functions ψ and ϕ defined on the cliques C and separators S respectively. Specifically, we require a joint distribution expressed as

$$P(V) = \frac{\prod_{C_i \in C}\psi(C_i)}{\prod_{S_j \in S}\phi(S_j)}. \tag{2.9}$$

Comparing expressions (2.8) and (2.9), it is easily shown that if all potentials $\phi(S_j)$ on separators S are set to unity, the potential representation on cliques is straightforward to achieve at initialisation. We have

$$\psi(C_i) = \prod_{V_j, PA(V_j) \in C_i} P(V_j \mid PA(V_j)).$$

If no node and its parents lie in that clique the initialising potential function on that clique can be considered to be unity. Now, since the model is decomposable we have a particular potential representation taking the form

$$P(V) = \frac{\prod_{C_i \in C} P(C_i)}{\prod_{S_j \in S} P(S_j)}, \tag{2.10}$$

where function P indicates the marginal distributions on the cliques and separators. From this representation the marginal distributions on any single node V_i may be easily obtained from the distribution on any clique which contains V_i,

$$P(V_i) = \sum_{C_j - V_i} P(C_j) \qquad where \;\; V_i \in C_j .$$

The L-S procedure is essentially concerned with systematic re-representation of the Bayesian belief network in terms of a clique tree and the probability distribution in terms of different potential representations on cliques and separators in order to allow efficient calculation of variables of interest. We will show the representation (2.9) will hold whatever evidence is received on the network and then describe the evidence propagation algorithm.

In general, the computational algorithm for making a graph triangulated, constructing a clique tree and initialisation can be described as following:

BEGIN
  FOR each node V_i DO
    IF (V_j and V_k are parents of V_i) AND (V_j and V_k are not connected)
    THEN connect V_j and V_k
drop directions on edges
assign number 1 to an arbitrary node
FOR i=2 to n DO
  BEGIN
    select the node having the largest set of previously numbered nodes
    IF there are more than one candidates
    THEN break ties arbitrarily
    assign number i to the selected node
  END
FOR i=n to 1 DO
  BEGIN
    recursively fill in edges between any two neighbours
    of V_i having lower ranks than V_i
    (including neighbours linked to V_i in previous steps)
  END
find all cliques
order the cliques by the highest numbered node in each clique
FOR each clique C_i DO
  determine separators S_i and parent cliques of C_i
choose a root clique C_R
FOR each other clique C_i DO
  BEGIN
    add a link between C_i and its parent clique
    assign the separator between them to the link
  END
FOR each clique C_i DO
  BEGIN
    set the potentials on C_i ($\psi(C_i)$) to unity
    FOR each member V_j of C_i DO
      IF (V_j and $PA(V_j)$ are in C_i) AND ($P(V_j|PA(V_j))$ has not been visited)
      THEN
        BEGIN
          calculate the potential $\psi(V_j)$ from the conditional probability tables
          $\psi(C_i)=\psi(C_i)*\psi(V_j)$
          mark the conditional probability $P(V_j|PA(V_j))$ visited
        END
  END
  FOR each separator S_i DO
    set the potentials on S_i ($\phi(S_j)$) to unity
END

2.4.2.2. Evidence Propagation

The implication of evidence can be propagated to any other node through the clique tree without any global supervision to prevent inconsistencies. Each clique can receive and pass on "messages" from its neighbours through separators, which lie between them [30,31]. We now have a joint distribution expressed in terms of potential functions on cliques and their separators, and the next step is to show that this representation will hold whatever evidence is received on the network. For any evidence $E=e$, $P(V-E \mid E=e)$ can be expressed in the form (2.8). This is because we always have

$$P(V-E \mid E=e) = \frac{P(V-E, E=e)}{P(E=e)} \propto P(V-E, E=e).$$

When evidence arrives, it is absorbed in cliques containing the node that has been observed and any potential not defined on the observed value is set to zero. There are two ways to propagate evidence: "distribute evidence" and "collect evidence". "Distribute evidence" is used when evidence from a single clique must propagate to the entire clique tree. "Collect evidence" is used when evidence from the entire clique tree must propagate to a single clique.

When evidence has been absorbed in more than one clique, it can be propagated to the entire clique tree as follows. A root clique is arbitrarily chosen in the clique tree. The root clique issues a message "collect evidence" to its neighbours, who pass it to their neighbours until it reaches the leaves of the tree. Each clique then collects evidence from those neighbours further from the root clique, using the fundamental operations: marginalisation, update and renew. Consider a clique C_k with neighbours $C_{W_1}, \ldots, C_{W_n}$ with separators $S_{W_1}, \ldots, S_{W_n}$ respectively. The potential function on C_{W_i} and S_{W_i} are $\psi(C_{W_i})$ and $\phi(S_{W_i})$.

Marginalisation will perform

$$\phi^*(S_{W_i}) = \sum_{C_{W_i} - C_k} \psi(C_{W_i}).$$

A new potential $\phi^*(S_{W_i})$ is calculated. Updating is carried out then

$$\psi^*(C_k) = \psi(C_k) \frac{\phi^*(S_{W_1})}{\phi(S_{W_1})} \cdots \frac{\phi^*(S_{W_n})}{\phi(S_{W_n})}.$$

Finally $\psi(C_k)$ is renewed with $\psi^*(C_k)$. It is important that in collecting evidence each clique must await the messages from its more distant neighbours before passing messages on towards the root.

When finally the root clique has collected evidence from its neighbours, it normalises its new potential for evidence distribution. In the procedure of distribution, the same message passing operation is

involved but working back through the tree. When this is complete all cliques and separators will hold their correct and updated marginal distributions. The current distribution for any node can be easily derived from these cliques or separators. The algorithm described above is a simplification of that presented in [28], and proofs that these operations lead to this conclusion may be found in [30].

As far as uncertain evidence is concerned, we can assume that there is a dummy node which influences our belief in observed variable V_j. This dummy node represents our observation (uncertain evidence). Let E' stand for relevant observation on variable V_j. We then assume that node E' is observed with certainty. It will change our belief in observed variable V_j to $P(V_j|E')$. The realisation of the dummy node passes the impact $P(V_j|\textit{relevant observation})$ to observed variable V_j. Assume that clique C_k has node V_j and the uncertain evidence concerning V_j is expressed as $P^*(V_j)$ by the user. The evidence absorption in clique C_k is carried out by

$$\psi^*(C_k) = \psi(C_k)\frac{P^*(V_j)}{P(V_j)}.$$

After the evidence absorption, we will use the same message passing technique developed to propagate uncertain evidence. That is, to call "collect evidence" followed by a call to "distribute evidence" from the root clique. It is important to know that the result of propagating uncertain evidence is consistent. This is because (i) we ask for consistency of observation; (ii) the propagation scheme is coherent.

2.4.3. A Comparison Between Pearl's and Lauritzen-Spiegelhalter's Approaches

The computation required by a probabilistic reasoning system should be "feasible" in that it can be completed in a "fair" amount of time, so that the system can be applied to solving a real problem. It is very useful to estimate computational complexity of these two approaches. The computational complexity for these algorithms has not been completely analysed in terms of the network topology [77]. Here we try to analysis the complexity in three cases: tree structure, general graph and highly dependent graph.

Tree Structure

Suppose that there are N nodes and L edges in the tree ($L=N-1$). The maximal values that a node may takes is M. The maximal number of children is K. When the underlying graph is a tree structure, we can apply Pearl's approach directly. Each node will perform updating on its π message and λ message, renewing its belief. In general, it needs M^2 operations to update its π message. To renew its λ message requires MK operations. The operation of computing a new belief is M. In addition, passing messages up and down also needs extra computations. It will not exceed $MK+M^2$, where MK is for sending a new message π to its children and M^2 is for sending new message λ to its parents. Therefore the total computation for propagating a piece of evidence by using Pearl's algorithm is less than

$$N(M^2+MK+M+MK+M^2)=N(2M^2+2MK+M) \sim O(LM^2).$$

To store π message, λ message and its belief, the space requirement of each node is $M^2+M+MK+M$. The total space requirement is

$$N(M^2+M+MK+M)=N(M^2+2M+KM) \sim O(NM^2).$$

The L-S algorithm has an upper bound of $3R+g\Theta$ elementary arithmetic operations, where R is the total size of the state space, g is the number of cliques and Θ is the size of the largest state space of a clique [28]. In the tree structure, each node has at most one parent and the tree is already a triangulated graph. No additional edge is needed. The two nodes of a edge form a clique. Therefore there are L cliques and each clique has only two nodes. The total state space is less than M^2L. The largest state space of a clique can not exceed M^2. So the complexity of propagating a piece evidence in the L-S algorithm stays less than

$$3M^2L+LM^2=4LM^2 \sim O(LM^2).$$

The operations are performed on cliques and separators in the L-S algorithm. We need to store all the marginal distributions on the cliques and the separators. The total space requirement is

$$LM^2+LM=L(M^2+M) \sim O(LM^2).$$

In the polytree structure, we assume that each node has at most S parent nodes. We can still apply Pearl's approach directly. The total computation for propagating a piece of evidence by using Pearl's algorithm is less than

$$N(M + KM + M^S + SM^{K+1} + KM^{S+1}) \sim O(NSM^{K+1} + NKM^{S+1}).$$

The space requirement of Pearl's approach has the similar function. However, additional edges are needed to marry parents of the polytree. The complexity of propagating a piece evidence in the L-S algorithm stays less than

$$3M^{S+1}N + LM^{S+1} \sim O(NM^{S+1}).$$

The space requirement is

$$LM^{S+1} + LM^S + LM^K \sim O(LM^{S+1} + LM^K).$$

In the case of tree structure, the complexities of both algorithms are linear functions of the size of trees. However, the maximal number of values a node may take also plays an important rule in this case. In a real problem, the maximal number of values a node may take is reasonably large. Pearl's algorithm is "tree" specific, the L-S algorithm is not. Therefore Pearl's algorithm may be more suitable than the L-S algorithm.

General Graph

By a general graph we mean that there are some loops in the graph, i.e. it is a sparse and irregular graph. More precisely, the number of edges is only linear in the number of nodes, whereas highly dependent graphs can have a quadratic number of edges. We have to instantiate variables in the loop cut set of the graph to render the remaining graph singly connected and thus amenable to Pearl's algorithm. We need to consider each possible combination of values of the loop cut nodes, however, as there are that many instances. Therefore the computational complexity is a function of the product of the number of possible values of the loop cut nodes and the size of the reduced singly connected graph. Suppose there are N_L loop cut nodes in the graph. The reduced graph requires C_R operations to be updated, which has similar complexity of tree structure to that discussed in the previous section. So its computational complexity in total is $C_R N_L{}^M$. The size of the loop cut nodes is the main contribution to the complexity. The space requirement is a function of $C_R N_L{}^M$.

On the other hand, the L-S algorithm can be applied directly to this kind of graph. Assume we have N_C cliques and the maximal number of nodes in a clique is M_C. So the complexity is less than $4N_C M_C{}^M$

($3N_C M_C{}^M + N_C M_C{}^M$). The computation of evidence propagation is a linear function of the number of cliques, but is exponential in the size of the largest clique in the graph. The maximal clique state size is the crucial element in complexity. The space requirement is a function of $N_C M_C{}^M$.

It is clear that the number of loop cut nodes is a crucial factor for Pearl's algorithm. On the other hand, the size of the largest clique in the L-S algorithm is the key factor. In general, the size of the largest clique in the general graph is not large, in particular, the graph is large and sparse. The L-S algorithm is expected to perform well. Therefore the L-S method seems better suited to graphs than Pearl's (which is better for trees). For example, the HUGIN system can perform diagnostic inference in under five seconds on the MUNIN network for neuromuscular disorders, containing about 1000 variables [58,56].

Highly Dependent Graph

This is a form of large broad and shallow graph, known as a dense graph. Typically, the number of edges of the graph is greater than one fourth of the maximum number of edges [78]. A child may have a lot of parents and a parent may have a lot of children with many intersecting loops. For example, two-level Bayesian belief networks [77] and QMR-BN belief networks [79]. In this case, the number of the loop cut nodes is large. The number of cliques is small, but the size of the clique may be very large. For these kind of networks, both the computation complexity and the space requirement are exponential functions of the number of nodes in the graph. Both algorithms are not flexible in such a case.

2.4.4. Approximate Approaches

Both Pearl's message passing algorithm and the L-S clique algorithm are called exact algorithms. In other words, they provide an exact solution. As discussed earlier, exact algorithms need to exploit the conditional independence in the network in order to provide an exact solution effectively. Although there has been significant improvement in these algorithms, they are effectively limited to problems of special structure or small size. Using standard methods drawn from the theory of computational complexity, researchers in the field have shown that the problem of probabilistic inference in Bayesian belief networks is difficult and almost certainly intractable. More generally, probabilistic reasoning using Bayesian belief

networks is known to be NP-hard† [57,80]. The NP-hard problems do not admit polynomial-time exact computational algorithms [81]. This strongly suggests that different strategies should perhaps be used to accomplish probabilistic reasoning in complex domains.

The development of approximate algorithms for probabilistic reasoning in Bayesian belief networks may be one of many possible solutions [32,59,71,77]. These approximate algorithms provide approximate answers, that is, the answer one gets is not exact but with a high probability that it is within some small distance of the correct answer. They have been shown to be valuable alternatives to exact methods for general analysis of large problems [79].

A Bayesian belief network defines the joint probability distribution space as the product of conditional probabilities of each node given its parents. There is an alternative way to express the joint probability distribution as a set of finite samples. The sample space is a set of state vectors. The number of state vectors in the sample space is called the sample size. The element of the state vector represents the state of each variable in the Bayesian belief network. The Bayesian belief network serves as simulated sample generators. Simulations can be conducted to generate these samples according to a given model. The probability of a value for a given variable can be computed by recording the proportion of time that the variable is assigned the value in a sample space. For example, the probability of V_i is estimated as:

$$P(V_i) = \frac{\textit{the number of state vectors in which } V_i \textit{ is present}}{\textit{the total number of state vectors in the samples}} . \tag{2.11}$$

This is a frequency interpretation of probability.

Gibbs sampling was described by Geman and Geman [82]. They have pointed out a close correspondence between Random Markov Fields and the Gibbs distribution, describing stochastic properties of systems with an additive energy function. The simulation scheme has been successfully applied to image restorations.

Bundy [15] suggested one such Monte Carlo approach for computing the probabilities of Boolean combinations of correlated logic variables, which he called the "incidence" calculus. The incidence calculus represents samples of the joint probability space as bit strings and defines axiomatised procedures

---

† A problem Π is NP-hard if there are exists some NP-complete problem Π' which would be solved by a polynomial time deterministic algorithm provided it is equipped with some routine to solve Π. So NP-hard problem is at least as hard as problems in the NP class.

for deriving new bit strings. More recently, the scheme of logic sampling [71] has applied incidence calculus to Bayesian belief networks. When applying the logic sampling algorithm to networks without evidence, sampling in each simulation starts from the root nodes and works down to the leaf nodes. The prior distribution of each root node is used to guide the choice of a sample value from the node's state space. Because Bayesian belief networks are acyclic, once any root node is sampled, it must leave the network with at least one new "root" node. This property insures that the process of sampling will be continuous until all nodes in the network are sampled.

However, logic sampling does not deal well with evidence. Logic sampling does not permit evidence nodes to be set to their known values, unless they just happen to be root nodes. For this reason, there is no way to account for evidence until the nodes corresponding to these observations are sampled. If they match the observations, the simulation is counted, otherwise, it must be discarded. Logic sampling may generate a large number of irrelevant samples [32]. Therefore, the complexity of pure logic sampling is exponential in the number of observed variables because all instantiations inconsistent with observed evidence are simply dropped from the samples. There are various enhancements to logic sampling. For example, likelihood weighting [83], and importance sampling [84]. Chavez and Cooper [85] explore hybrids of logic sampling and Gibbs sampling.

2.4.4.1. Stochastic Simulation Algorithm

To overcome the disadvantage of the logic sampling algorithm, an approach called stochastic simulation has been proposed [32]. It has been shown that the stochastic simulation can be viewed as a sampling from the Gibbs distribution [86]. In fact, the stochastic simulation modifies the logic sampling by adding a preprocessing step to each simulation. The preprocessing step involves each node performing local computations in order to determine a probability distribution for sampling. Without loss of generality, we assume all the variables are binary. For convenience, we will use upper case letters to represent variables and lower case letters to represent their particular states, for example, t and $\neg$t, corresponding to *true* and *false*, are two possible states of variable T. The stochastic simulation algorithm can be described as follows [87,88]:

```
BEGIN
    initialise the state of node with evidence
    WHILE no query is raised DO
    BEGIN
      FOR each variable V_i in the causal model DO
      BEGIN
          compute the conditional probability P(v_i | X_Vi) and P(¬v_i | X_Vi)
          (where X_Vi is the state of all variables except V_i in the model)
          consult a random number generator that produces 1 to 0 by a
               ratio of P(v_i | X_Vi) to P(¬v_i | X_Vi)
          choose a state for the variable V_i according to the outcome of the generator
      END
      END
      compute the marginal probability using the expression (2.11)
END
```

When we compute the conditional probability $P(v_i \,|X_{V_i})$ and $P(\neg v_i \,|X_{V_i})$, we simplify the computations by considering conditional independence reflected in the Bayesian belief network. Let us consider $P(V_i \,|X_{V_i})$ now,

$$P(V_i \,|X_{V_i}) = \frac{P(V_i, X_{V_i})}{P(X_{V_i})} = \frac{P(V_1, V_2, \cdots, V_m)}{P(X_{V_i})} \propto P(V_i, X_{V_i}) \,. \tag{2.12}$$

$P(X_{V_i})$ is a constant independent of V_i. We can calculate the joint probability distribution $P(V_i, X_{V_i})$ on the network

$$P(V_i, X_{V_i}) = \prod_{l=1}^{m} P(V_l \,|PA(V_l))$$

$$= P(V_i \,|PA(V_i)) \prod_{j=1}^{n} P(C_j \,|PA(C_j)) \prod_{k=1}^{m-n-1} P(V_k \,|PA(V_k)) \,, \tag{2.13}$$

where m is the number of nodes in the network, C_j is a child node of V_i, n is the number of V_i's children and V_k is a remaining node in the network. $\prod_k P(V_k \,|PA(V_k))$ in the expression (2.13) is also a constant independent of V_i. Finally, we can get (from the expressions (2.12) and (2.13))

$$P(V_i \,|X_{V_i}) = \alpha \, P(V_i \,|PA(V_i)) \prod_{j=1}^{n} P(C_j \,|PA(C_j)) \,, \tag{2.14}$$

where α is a normalising constant.

We only need to consider V_i's parents, children and children's parents in order to compute conditional probability $P(V_i \,|X_{V_i})$ in each simulation. Therefore, the computation involves only a small number of nodes.

Having calculated $P(v_i \mid X_{V_i})$ and $P(\neg v_i \mid X_{V_i})$, the state of V_i is sampled by using a random number generator and comparing its outcome with the computed conditional probability. If the outcome is less than the calculated conditional probability $P(v_i \mid X_{V_i})$, the new state of V_i is *true*, otherwise it is *false*. The simulation process is repeated in this manner.

However, the simulated probability may be sensitive to conditional probabilities. Consider as a simple example, the network with two binary variables shown in Figure 2.5. Given that the prior probability of node X is 0.5 and the symmetry of the conditional probabilities (i.e. P(y|x)=P(¬y|¬x)) linking the two nodes, it is apparent that the marginal probability of Y is also 0.5. Five simulation runs are conducted for each of the conditional probabilities. The sample sizes are taken as 1000, 5000 and 10000. The results of experiments on different conditional probabilities are presented in the following three tables. These results show that the performance of the stochastic simulation algorithm are poor when the conditional probabilities are approaching extremes.

Figure 2.5. A Simple Example

P(y|x)	Simulation Size=1000					Average of Error |estimated value - 0.5|
	Run 1	Run 2	Run 3	Run 4	Run 5	
0.50	0.494	0.513	0.501	0.500	0.501	0.0042
0.55	0.489	0.509	0.503	0.526	0.499	0.0100
0.60	0.502	0.500	0.487	0.518	0.519	0.0104
0.65	0.495	0.488	0.485	0.506	0.513	0.0102
0.70	0.501	0.483	0.513	0.508	0.520	0.0118
0.75	0.503	0.486	0.482	0.517	0.502	0.0108
0.80	0.507	0.478	0.493	0.515	0.513	0.0128
0.85	0.523	0.479	0.500	0.513	0.518	0.0150
0.90	0.539	0.479	0.468	0.518	0.482	0.0256
0.95	0.528	0.450	0.399	0.558	0.471	0.0532
0.975	0.511	0.538	0.457	0.585	0.502	0.0358
0.99	0.549	0.626	0.455	0.355	0.291	0.1148
0.999	0.519	0.417	0.148	0.105	0.475	0.1748
0.9999	1.000	1.000	1.000	1.000	0.664	0.4328

P(y|x)	Simulation Size=5000					Average of Error |estimated value - 0.5|
	Run 1	Run 2	Run 3	Run 4	Run 5	
0.50	0.5120	0.5062	0.5114	0.5074	0.4892	0.00956
0.55	0.5108	0.5048	0.5076	0.5088	0.4886	0.00868
0.60	0.5044	0.5012	0.5130	0.5094	0.4900	0.00760
0.65	0.5024	0.4985	0.5160	0.5048	0.4886	0.00722
0.70	0.5008	0.4970	0.5210	0.5078	0.4864	0.00924
0.75	0.4988	0.4970	0.5148	0.5018	0.4878	0.00660
0.80	0.5012	0.5002	0.5138	0.5036	0.4834	0.00708
0.85	0.5054	0.5056	0.5180	0.5110	0.4714	0.01372
0.90	0.5012	0.4960	0.5258	0.5146	0.4756	0.01400
0.95	0.4818	0.4798	0.5066	0.5334	0.5064	0.01696
0.975	0.5276	0.5162	0.4776	0.5210	0.4898	0.01948
0.99	0.4286	0.4492	0.4606	0.5808	0.5302	0.05452
0.999	0.2502	0.7850	0.6780	0.8472	0.5636	0.22472
0.9999	0.9320	1.0000	0.5390	1.0000	0.2196	0.35028

P(y|x)	Simulation Size=10000					Average of Error |estimated value - 0.5|
	Run 1	Run 2	Run 3	Run 4	Run 5	
0.50	0.5088	0.4982	0.4935	0.5031	0.5067	0.00538
0.55	0.5062	0.4986	0.4942	0.5034	0.5068	0.00472
0.60	0.5071	0.4996	0.4944	0.5022	0.5080	0.00466
0.65	0.5066	0.4966	0.4966	0.4998	0.5068	0.00408
0.70	0.5089	0.4970	0.4974	0.4993	0.5081	0.00466
0.75	0.5058	0.4948	0.5008	0.4960	0.5072	0.00460
0.80	0.5069	0.4935	0.5038	0.4974	0.5073	0.00542
0.85	0.5117	0.4912	0.5079	0.5039	0.5105	0.00856
0.90	0.5018	0.4951	0.5082	0.5022	0.5108	0.00558
0.95	0.4917	0.5196	0.4828	0.4881	0.5189	0.01518
0.975	0.4954	0.5052	0.5163	0.5106	0.5010	0.00754
0.99	0.4515	0.5547	0.4995	0.4530	0.5479	0.03972
0.999	0.4995	0.7626	0.5651	0.3549	0.7716	0.16898
0.9999	0.4660	0.3549	0.1563	0.5529	0.5465	0.12444

2.4.4.2. The Problem of Slow Convergence

It is a reasonable thought that the sample space generated in this way will statistically cover all possible states in coherent proportions. However, it is not true for certain types of causal models, where some probabilities are set to 1 or 0. The problem of slow convergence will be illustrated with a simple example in the next section. For simplicity, consider three variables, named S, T and E. Without loss of generality, we will assume that variables are binary. For example, t represents T=*true* and ¬t represents T=*false*. A Bayesian belief network representation is shown in Figure 2.6 and the corresponding conditional probability table is given in Table 2.1. Note that some of the conditional probabilities defined are 1 or 0. In fact, S and T can be thought as input of "OR gate" and E output of "OR gate".

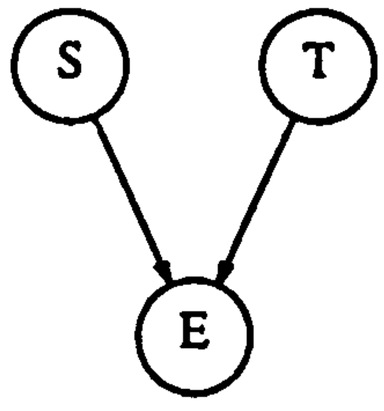


Figure 2.6. An Example of Logic Relations - OR Gate

P(s)=0.2	P(t)=0.4
P(e|t,s)=1	P(e|t,¬s)=1
P(e|¬t,s)=1	P(e|¬t,¬s)=0

Table 2.1. Conditional Probability Table

Now we will use the stochastic simulation algorithm described above to calculate the marginal probability of each variable. Suppose that there is no evidence at the moment. First of all, the initial state is chosen arbitrarily or by some appropriate criteria. The next step is to generate a finite sample of possible states of three variables at random from the causal model and the initial state. Let V_i^* denote the next possible state of variable V_i in a simulation. From the expression (2.14), we get

$$P(S^* \mid X_S) = \alpha_1 P(S)P(E \mid S, T), \tag{2.15}$$

$$P(T^* \mid X_T) = \alpha_2 P(T)P(E \mid S, T), \tag{2.16}$$

where α_1 and α_2 are two normalising constants. When the states of S and T are known, the state of E will be determined. It is interesting to note that there are some logically impossible states in the sample generation. These states are (s,t,¬e), (¬s,t,¬e), (s,¬t,¬e) and (¬s,¬t,e). For example, if T and S are in the state t and s respectively, E will definitely be in the state e because of P(e|t, s)=1.

Suppose the initial state is (¬s, ¬t, ¬e) and we sample variables in a order of S, T and E.

Processing S: by using expression (2.15), we have

$$P(s^* \mid X_S)=\alpha P(s)P(\neg e \mid s, \neg t)=0,$$

$$P(\neg s^* \mid X_S)=\alpha P(\neg l)P(\neg e \mid \neg s, \neg t)=\alpha 0.8.$$

Normalising these two conditional probabilities, we have α=1.25 so that $P(s^* \mid X_S)=0$ and $P(\neg s^* \mid X_S)=1$.

The state of S is set to ¬s, which is the same as its previous state.

Processing T: the expression (2.16) is applied:

$$P(t^* \mid X_T) = \alpha P(t) P(\neg e \mid \neg s, t) = 0,$$

$$P(\neg t^* \mid X_T) = \alpha P(\neg t) P(\neg e \mid \neg s, \neg t) = \alpha 0.6 .$$

Once again, we get $\alpha = 5/3$, $P(t^* \mid X_T) = 0$ and $P(\neg t^* \mid X_T) = 1$. In this case, the state of T is $\neg$t.

Processing E:

$$P(e^* \mid X_E) = P(e \mid \neg s, \neg t) = 0 ,$$

$$P(\neg e^* \mid X_E) = P(\neg e \mid \neg s, \neg t) = 1 .$$

E stays in the state $\neg$e.

The simulation continues in this way. However, the state of each node does not change no matter how many simulations we do. The estimated marginal probability of each node being *true* is always zero.

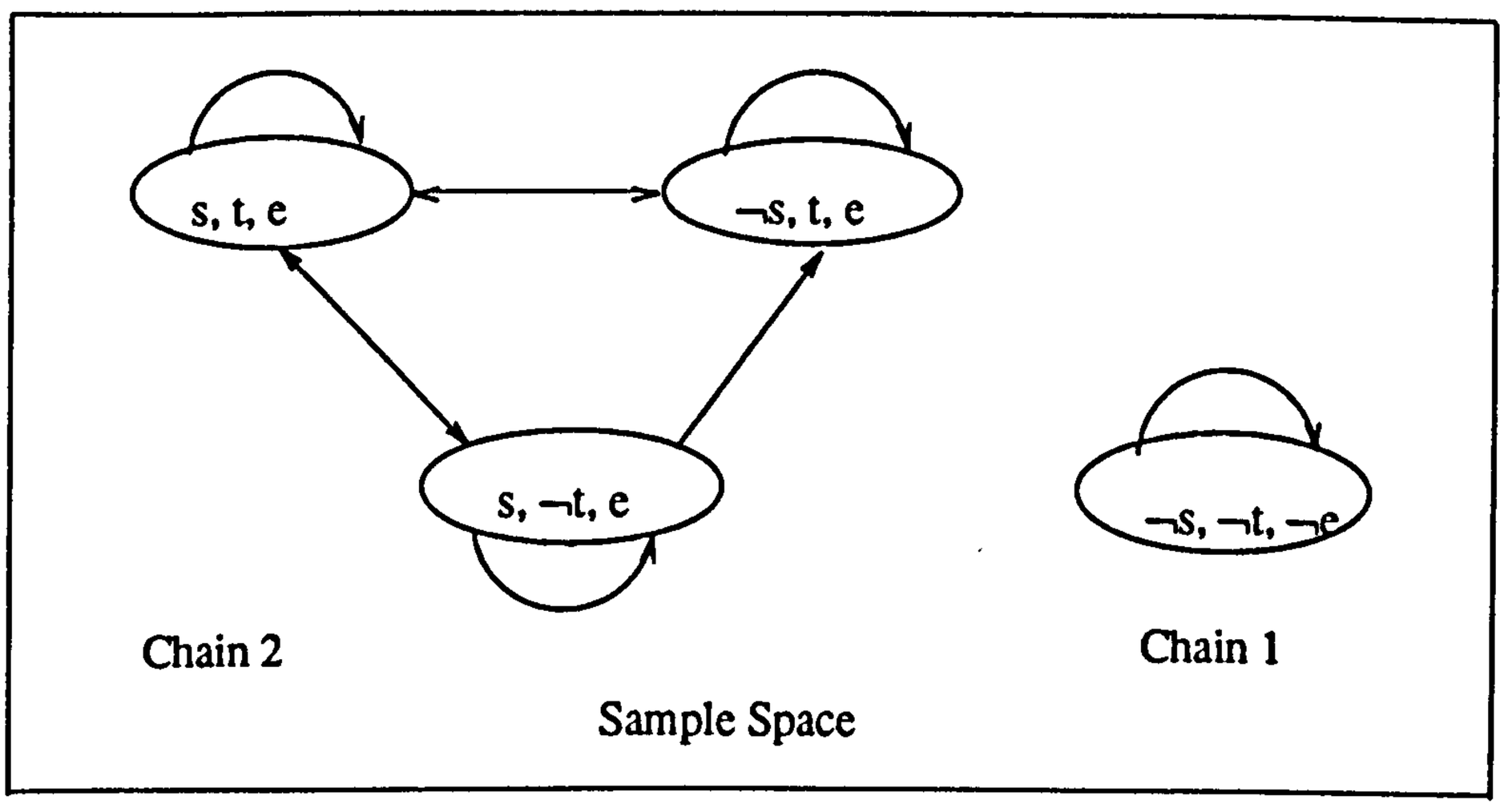


Figure 2.7. Possible State Transformations

If we start the simulation with any one of these states (s,t,e), (¬s,t,e) and (s,¬t,e) the states of three variables can change among these states, but they can not reach the state (¬s,¬t,¬e). The possible state transformation of our example is illustrated in Figure 2.7. It can be seen from Figure 2.7 that the sample space is divided into two isolated chains, Chain 1 and Chain 2, because there are some conditional probabilities, 1 or 0 in the example. It is impossible that the sample space generated by the stochastic simulation algorithm will cover all four possible states, see Appendix B.

In Chain 1, there is a state (¬s,¬t,¬e). Chain 2 consists of three states: (s,t,e), (s,¬t,e) and (¬s,t,e). The states can be changed inside the chain, but can not go to the other chain. The final results will tend to be P(s)=5/13, P(t)=10/13, and P(e)=1, see Table 2.2.

Node	Stochastic Simulation				Expected Value
	1000	5000	10000	50000	
s	0.342	0.357	0.3827	0.3836	0.3846
t	0.806	0.7988	0.7667	0.76934	0.7692
e	1.0	1.0	1.0	1.0	1.0

Table 2.2. Simulated Results of the Small Example

However the correct marginal probabilities for the variables are: P(s)=0.2, P(t)=0.4 and P(e)=$\sum_{S,T} P(e,S,T)$=0.52. The results of applying the stochastic simulation algorithm, no matter which initial state we start with, will not approach the exact values even for infinite simulations. A detailed discussion of the convergence problem is also presented in Appendix B.

The stochastic simulation algorithm degrades dramatically as the probabilities themselves approach 0 and 1 [89]. The distribution of generated samples does not necessarily correspond closely to the true distribution of outcomes. This behaviour is a result of the tendency for local areas in the network to become fixed through many simulation cycles rather than to mix readily in a random fashion according to their true probabilities. Several network-manipulation techniques that may lead to more efficient simulation of a given Bayesian belief network have been proposed to convert some networks into formats that are computationally more efficient for simulation [83,89]. It has been shown that the stochastic simulation algorithms based on "Markov" transitions are critically sensitive to numerical aspects of the models, particularly when parts of a model are close to deterministic [57,89]. A theoretical convergence analysis to a modified stochastic simulation algorithm is presented in [57]. However, convergence problems are the main research topics in stochastic simulation.

2.4.5. Related Reasoning Algorithms

Reasoning algorithms have also been developed for influence diagrams [27,90] and belief functions [66,68]. Shachter focuses on the computation of the posterior distribution given the evidence of a single

variable (or some of the variables) in the influence diagram, where the node also could be a value node and a decision node [27,90]. A value node represents the objective to be maximised in expectation by the decision analysis. A decision node represents a variable whose value is chosen by the decision maker. Formal statistical theory says nothing about which direction the arrows should go in, as long as the network has no directed cycles. Using Bayes' theorem, it is possible to revise edges if we change the corresponding probability distributions accordingly. For example, P(E|H)P(H)=P(H|E)P(E). So we can have the following graphs which have the same probability distribution P(H,E) [91].

Based on this observation, evidence propagation is implemented by the basic operations of eliminating a node, revising the edges showing conditional dependence and updating the distributions within the nodes. These operations transform one influence diagram into another, without changing the underlying joint distribution for the variables of interest. In order to solve the inference problem P(X|E), a node without any child node which is not an element of X, E can be eliminated. Starting from evidence E, a sequence of operations of edge reversal and updating the distributions within the nodes, and taking expectations over nodes to eliminate them can be performed. The process continues in this way. Finally, we have only two nodes E and X in the influence diagram. The posterior probability P(X|E) is then determined.

Shenoy and Shafer consider a belief function which is a set function rather than a point function and a belief function propagation in Bayesian belief networks [66,68]. The graphical structure is converted into a hypertree, which explores qualitative independence [92,93]. The evidence can be decomposed into independent items, each involving only a few variables. Each item of evidence is represented by a belief function. These belief functions can be combined by Dempster's rule, which is multiplicative in terms of the commonality functions [13]. This rule is appropriate for combining belief functions that are based on independent items of evidence. The result is a belief function representing the total evidence. Local computation is possible in the propagation of belief functions in the hypertree. The propagation scheme is very similar to that of L-S but the messages passed are always belief functions.

2.5. Concluding Remarks

Bayesian inference has been used in a number of expert systems [2,7,38]. Nevertheless, the early development of expert systems faced two kinds of difficulties. First, there is the representational problem, that is, how to structure and encode the knowledge of human experts into a coherent probabilistic form. Second, there is the inferential problem, that is, the issue of the computational tractability of algorithms for probabilistic inference and decision making. Probability theory has to be weakened in one way or another for real applications. However, it is an open question as to what extent that the theory should be relaxed to achieve a computationally feasible but effective way of handling uncertainty for real applications. These difficulties led most AI researchers to turn away from exact probabilistic representation [9,94].

The Bayesian belief network provides knowledge engineers the flexibility to specify and reason about dependencies. The use of Bayesian belief networks in probabilistic expert systems can resolve the representational problem, found to be so severe in the PROSPECTOR model [12]. In general, we are interested in how the realisation of some variables affects the other variables in probabilistic expert systems. Bayesian belief networks can be used to guide the efficient computations of required probabilities in probabilistic expert systems. The key to computational efficiency for probabilistic reasoning in Bayesian belief networks is to take advantage of conditional independencies reflected by the network topology, and to find ways of propagating the impact of new evidence locally without having to calculate the entire joint distribution explicitly. Together with methods derived from Bayes' decision theory this representation offers a consistent and computationally manageable means of handling uncertainty in expert systems.

It is interesting that for the exact algorithms, the feature of the network that determines performance is the topology, but for the approximate algorithms, it is the conditional probability. More general classes of Bayesian belief networks have eluded efforts to develop efficient inference algorithms. It is fruitless to search for a single computational algorithm for handling all Bayesian belief networks [95]. The choice of a reasoning algorithm corresponding to the topology of a Bayesian belief network is crucial for any real world application. It is important to match the given domain with the inference algorithm that will be most efficient given the specific properties of the domain model.

An ideal probabilistic reasoning system should be able to combine the advantages of the above algorithms, i.e. (1) it should be as general as possible; (2) it should be efficient in both time and space aspects, that is, the underlying space has to be decomposed into some subspaces to avoid a possible exponential explosion; (3) it should be easy to implement and use. Based on these considerations, a general purpose graphical environment for constructing and processing probabilistic knowledge systems using Bayesian belief networks is designed and developed in the next chapter.

Chapter 3

PROBABILISTIC REASONING EXPERT SYSTEM SHELL (PRESS)

As discussed in chapter 2, Bayesian belief networks are characterised as tools for constructing coherent probabilistic representations of uncertain expert opinions. From the knowledge engineering angle, they provide a suitable framework both for knowledge elicitation and for evidence propagation involved in probabilistic reasoning.

From the user's point of view, a graphical environment for probabilistic reasoning using Bayesian belief networks has some advantages. Firstly, this would replace the menu driven interface with something more convenient. Secondly, a graphical display would make the probabilistic reasoning much easier to understand and much more intuitive. Thirdly, it also makes it easier for the user to direct the system towards the desired conclusions whenever it strays. Finally, the criticisms of probabilistic models of uncertainty are overcome by an intelligent graphical interface that incorporates explicitly conditional independence [96]. Based on these considerations, a probabilistic reasoning expert system shell is desirable.

In this chapter, we attempt to exploit both graph-theoretic and probabilistic ideas for efficient design and implementation of a probabilistic reasoning expert system shell, called PRESS [33]. Section 3.1 reviews some existing computational systems, designed for analogous purposes. The design guidelines for PRESS are described in section 3.2. Section 3.3 introduces the graphical interface. In section 3.4 the design and implementation of PRESS will be described. Three reasoning algorithms discussed in chapter 2 are investigated and evaluated using PRESS in section 3.5. PRESS is illustrated by an artificial example in the field of forensic science [97] in section 3.6. Finally, the main characteristics of PRESS are summarised in section 3.7. The software aspects of PRESS are presented in Appendix G.

3.1. Related Computational Systems

Several systems have been developed for probabilistic reasoning using Bayesian belief networks [96,98,99,58,56,100]. Here we give a short overview of some of them. Unfortunately, no detailed implementation descriptions of these systems are available.

DAVID [96] is a decision network processing system that runs on an Apple Macintosh and provides operations for expected-value decision making and sensitivity analysis. The final output of DAVID consists of tables that indicate preferred decisions and expected values for every possible instantiation of the observed chance variables. DAVID is an example of a stand-alone system using an exact algorithm. It provides a simple graphical interface, but it is oriented to single decision problem solutions only. As such, DAVID's paradigm does not represent optimally large diagnostic problems that have dozens or even hundreds of observations.

KNET [98] is another well-known environment for constructing probabilistic knowledge bases within the axiomatic framework of decision theory. It contains an interface that tightly integrates HyperCard, a hypertext authoring tool for the Apply Macintosh computer. It could encompass hypermedia in the near future [98]. KNET uses Pearl's message passing algorithm. The system handles decision nodes by using a technique that transforms any Bayesian belief network algorithm into an influence diagram algorithm[101]. KNET differs from DAVID in the following respects: (1) it optimises the final output; (2) it provides multiple interfaces; (3) it has an open architecture.

DELIEF [99] is an interactive system that allows the design of belief function arguments based on Dempster-Shafer (DS) theory via a simple graphical inference [66,102]. The user can construct a graph where nodes represent variables and edges represent relations among these variables. The theory behind this system is the development of DS theory on Markov trees [102,68]. It demonstrates a possible extension to Pearl's singly connected tree.

MUNIN [58] is a medical expert system for interpretation of electromyographic findings and diagnosis of muscle and nerve diseases. It is based on Bayesian probability theory and much of the medical knowledge in the system is represented in a causal probabilistic network. The network has roughly three levels: disease level, pathophysiological level and finding level. The levels are linked by causal relations

and links are expressed as conditional probabilities. The inference method used is an adaptation of Pearl's message passing algorithm [25,70]. The network is modified in order to get rid of loops.

HUGIN [56] is a shell system for building Bayesian belief universes for research expert systems. In fact, the system is a development based upon the MUNIN system. A simplification of the L-S method [28], that is, an algebra of Bayesian belief universes, allows propagation of evidence in an acyclic multiply connected network [31]. Although different terms are used in the system, a cliques tree is established as a data structure. Propagation of evidence is achieved by two basic operations: distribution evidence and collect evidence. In addition, the shell is menu-driven and most operations are carried out by mouse selection. The package is written in the C programming language and uses the X11 window system. HUGIN runs on a SUN platform. It provides a language for network definition and an easy graphical interface to run the probability propagation and to monitor posterior beliefs. No facility to deal with continuous variables and control are provided. Basically, it contains three main parts: an editor for causal networks and conditional probability tables; a compiler which transforms the network into a suitable data structure; and a runtime system which provides facilities for entering and propagating evidence.

ERGO [100] is a Bayesian belief network development environment. It provides the user with the power of an Apple Macintosh to develop belief networks with software as familiar as a drawing program and a spreadsheet. ERGO consists of a graphical drawing environment for creating a network, a probability editor for quantifying the associations among variables, and a probability engine for querying a belief-network model. ERGO can be embedded within a wide variety of programs to provide the real-time intelligence required for classification, diagnosis, and constraint-satisfaction problems.

In conclusion, the following aspects are identified as the important features of computational systems: a user-friendly graphical interface; powerful algorithms for evidence propagation; an efficient system for Bayesian belief networks storage and retrieval; an open architecture for possible extensions. The propagation algorithm is a major distinguishing feature, because it can either provide the exact posterior probability distribution of the Bayesian belief network, or approximate it. The design of PRESS will emphasis these issues in the next section.

3.2. Design Guidelines

PRESS combines normative probabilistic modelling techniques with a front end that offers the flexibility and expressive power of a graphical environment. Perhaps more importantly, PRESS clearly separates the design of a domain-specific user interface from all other aspects of the system. Several issues are addressed in the design of PRESS.

First, the expert system shell should be an interactive system that allows the design of models and performance of uncertain reasoning via a simple graphical interface. The shell should be menu-driven, with most operations carried out by mouse selection. Buttons, icons, menus, scrollbars and mouse-sensitive screen objects streamline the construction and validation of the knowledge base. The user-friendly environment is made possible by using tools provided by the particular system, the Sun Workstation. The use of the facilities (windows, mouse, menu, etc) makes a more amicable user interface. This improves the investigation of the problem at hand and allows the user to modify the model relatively easily.

Second, the shell should be able to integrate two different probabilistic reasoning mechanisms, the exact algorithm and the approximate algorithm, into a common computational framework. The user can select appropriate algorithms depending on the problem at hand.

Third, the results should be presented in various formats, including text, graphical and numerical. Whenever possible both numerical and graphical displays of the data and statistics are offered to the user. As far as probability distributions are concerned, they can be displayed in various formats, including numerical and graphical. For a particular node, a small popup window is used to display the current probability distribution of this node in more detail. A bar chart is displayed to give an overall view of all unobserved nodes in a graphical format.

Fourth, as far as programming is concerned, an object-oriented programming style would in this case, be desirable. It is clear that nodes are objects. Each node in the graph is internally represented as an *object*, which is the basic knowledge unit. When the user adds a node to the graph, an object is created. Each node/object has several properties: its name, the possible values of the variable, the topological relationships between the node and its neighbours (parents and children), and a conditional probability table.

Fifth, the irrelevant details of evidence propagation and belief updating should be hidden. The user never need observe the details of uncertainty management in PRESS. However, a help facility is available for the user and by means of this feature, the user is presented with a brief description of any menu items represented. PRESS should provide control facilities to help the user understand and control probabilistic reasoning.

Sixth, PRESS should be an efficient system for Bayesian belief network storage, retrieval, and communication capability with an external environment.

Finally, PRESS should be an open architecture for probabilistic expert systems. The system should be flexible and allow us to develop other functional modules using the same basic architecture.

3.2.1. Screen Design

Based on the design guidelines mentioned previously, the screen design of PRESS is presented in Figure 3.1. The screen is divided into four windows: *a control panel*, *a main menu selecting window*, *a drawing board*, and *a display window*.

The Control Panel (area A)

The upper window shows the function buttons and provides some basic operations. For example, storing and retrieving a Bayesian belief network and corresponding conditional probabilities in a user-defined name, which will be referenced for subsequent loading. Messages are also displayed to indicate where we are and what we are doing. Moreover, a screen dump option *print* is provided to give the user a hardcopy of the screen image. A *help* facility is available for directing the user to use this system and providing help at any stage if required.

The Main Menu Selecting Window (area B)

This is for displaying and selecting the main menu. When the user presses the right mouse button inside this window, the main menu will be displayed, see Figure 3.1. The main menu includes *Knowledge Acquisition*, *Exact Algorithm*, *Stochastic Simulation*, *Displaying Probability* and *Control Facilities*. Note that *Knowledge Acquisition*, *Exact Algorithm*, *Stochastic Simulation*, and *Control*

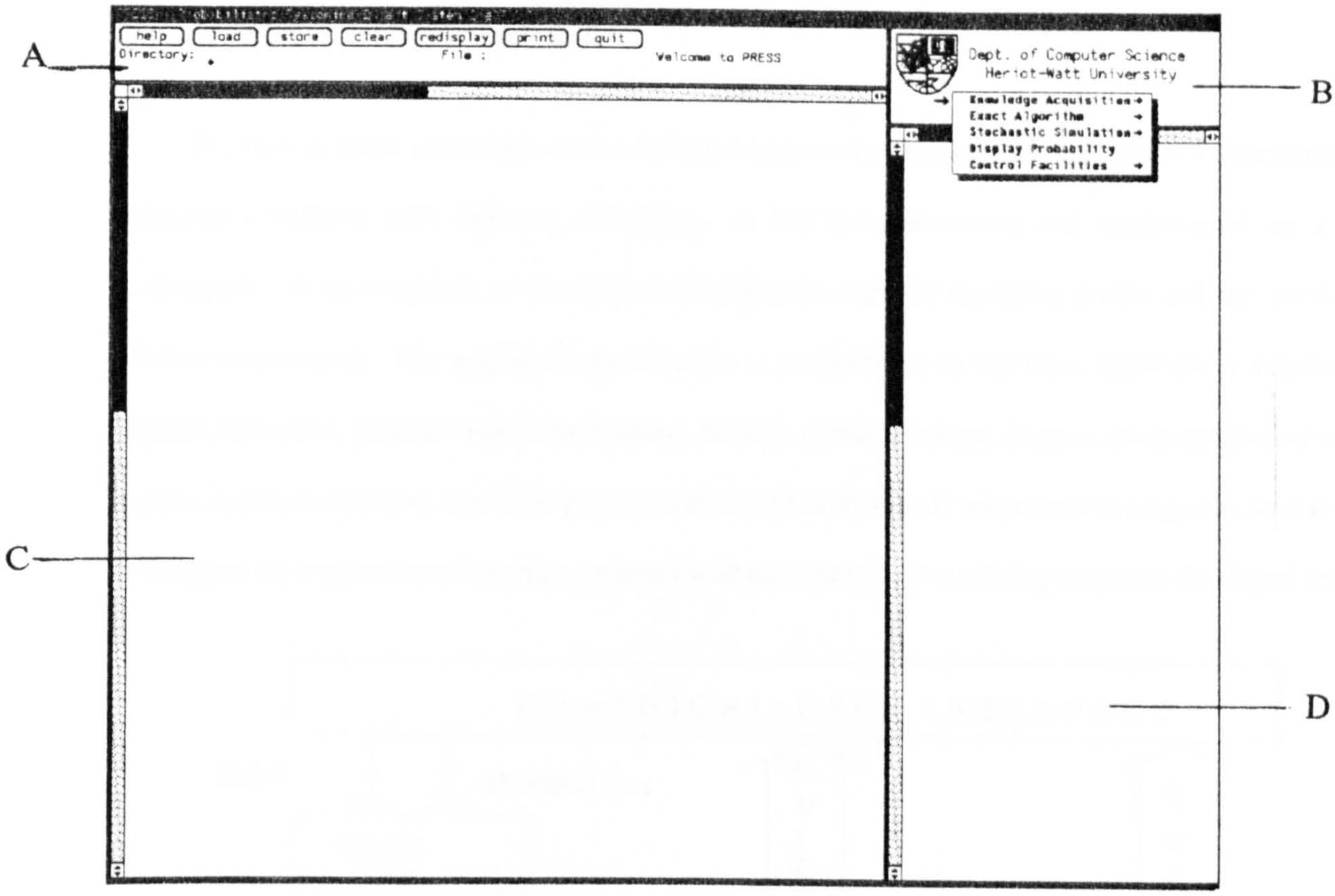


Figure 3.1. Screen Design of the System PRESS

Facilities are pull-right menus. Their menu items will be discussed later.

The Drawing Board (area C)

The drawing board is the widest window. It represents the Bayesian belief network editing window. It is a mouse-sensitive window for constructing and modifying a knowledge base, that is, a Bayesian belief network and conditional probabilities, and entering evidence. It is the main window where the user interacts with the shell system.

The Display Window (area D)

The display window (the right hand window) is dedicated to the visualisation of interesting information. It is used for displaying overall marginal probabilities and some measurements used in control facilities. Different display formats are used to help the user.

3.2.2. Shell System Architecture

The shell is menu and mouse driven for construction of expert systems in domains characterised by dependence relations with inherent uncertainty. It has been developed and implemented on a Sun workstation. All the programs were written in C using the SunOS† operating system and the SunView† window environment. The graphical user interface is programmed in SunView. SunView is a system to support interactive, graphics-based applications running within windows. It gives good integration of the mouse, buttons and menus. SunView programs are called within the C programming language, so it is easy to integrate the graphical user interface program with our control and reasoning programs developed earlier.

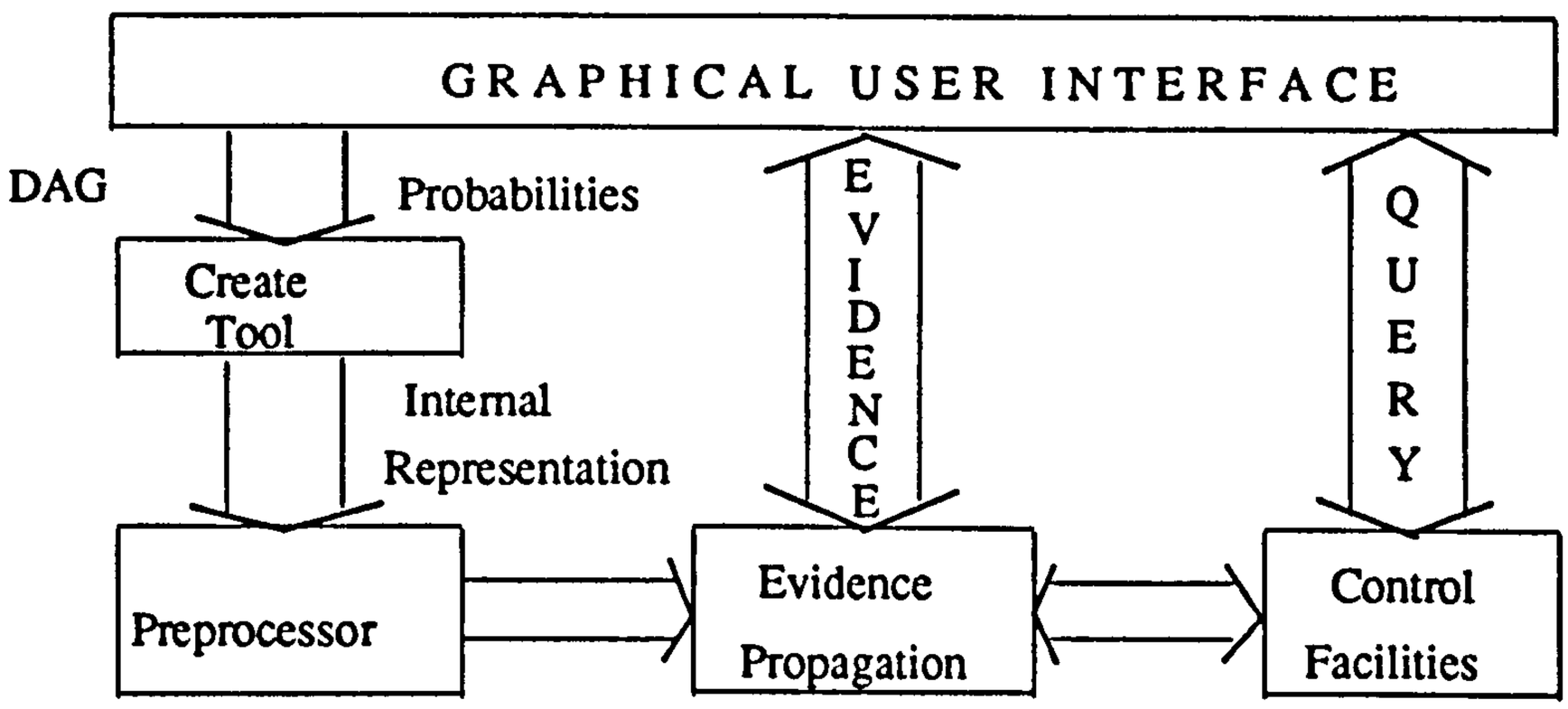


Figure 3.2. Basic Architecture of the System PRESS

The basic architecture of the shell system is depicted in Figure 3.2. It contains four main parts: *create tool*, *preprocessor*, *evidence propagation* and *control facilities*. The create tool (model construction) is used to construct and manipulate Bayesian belief networks and conditional probability tables. This tool is menu and mouse driven, and the user creates interactively the graphical representation of the domain knowledge. The preprocessor will deal with the domain knowledge by using different inference techniques for different structures. The input information, the structure and conditional probability tables, are processed and transferred into suitable internal representations for evidence propagation. All these have been implemented in the background. Evidence propagation allows the user to enter and propagate evidence in the structure and update belief in the light of evidence. The control facilities, given the measurement of the node of interest, help the user to control and understand the probabilistic reasoning

† SunOS and SunView are trademarks of Sun Microsystems, Incorporated.

process. All these measurements are displayed graphically and numerically on the screen. In the rest of this section, we address the problems of model construction and control facilities.

3.2.2.1. Model Construction

In this procedure, both qualitative and quantitative knowledge can be provided by the domain experts through the "create tool". Qualitative knowledge is represented as a directed acyclic graph. The nodes of the graph are random variables with finite values and direct influences of a node V_i are "parents" of V_i in the graph. The internal representation, a dependency matrix, is then set up based on these relations. Quantitative conditional probability distributions for each node V_i, given each configuration of its parent nodes, are needed, based on the dependency matrix. Conditional probability tables are then attached to each node for the Preprocessor.

3.2.2.2. Control Facilities

The PRESS system takes advantage of graphical knowledge representation. With this knowledge of dependence and information theory, we can measure the strength of relationships among any variables. Therefore, we can provide control facilities of probabilistic reasoning for the user, find the most influential evidence for a belief and order questions concerning a node of interest.

A Kullback-Leibler distance (KLD) in information theory is a suitable measurement [28,103]. For two discrete nodes U and W, their information measure (*IM*) is calculated using the following expression [103]:

$$IM(U,W) = \sum_U \sum_W \log\left[\frac{P(U,W)}{P(U)P(W)}\right] P(U,W)$$

$$= \sum_U \sum_W \log\left[\frac{P(U|W)}{P(U)}\right] P(U|W)P(W).$$

If W is of interest, but not observed, it is valuable and important to know what sort of questions concerning nodes to ask and in which order. If we know these, we can consider the consequence of each question and make optimal decisions. In other words, we can make a tradeoff between the cost of getting information and the level of uncertainty. The value of $IM(U,W)$ is interpreted as, how much information

U contains is relative to W. The bigger the value of $IM(U, W)$, the more relevant U is to W. We can rank possible questions according to the values of IM and the cost of each question. The information measurement of each node is calculated and displayed. Then question orders concerning unobserved nodes are sorted out, based on these values, and presented to the user for reference.

One important feature of expert systems are their ability to explain. PRESS can measure the probabilistic contribution that each piece of evidence provides to the final conclusions, known as influential findings. These contributions can also be measured by the Kullback-Leibler distance [28,103]:

$$I(V_j: E_n \mid E_1, ..., E_{n-1}) = -\sum_{V_j} \log \left[\frac{P(V_j \mid E_1, ..., E_{n-1})}{P(V_j \mid E_1, ..., E_n)} \right] P(V_j \mid E_1, ..., E_n),$$

where V_j is a node in the network and E_i is ith observed evidence. The value of I above shows that to what extent the presence of evidence E_n makes impact on our belief in V_j. Therefore, it is very useful for forming interpretations and showing how a conclusion is reached. In our system, both graphical and numerical formats are used to display the measurements. We propose that influential findings can be expressed as

Finds: (text)

Change of Probability: (both numerical and graphical formats)

Influences of Findings: (numerical format)

3.3. Working with PRESS

Before considering an example session with PRESS, let us briefly outline how the user works with the system.

(1) The user constructs a directed acyclic graph with each node representing a random variable, and the arcs signifying the existence of direct influences between the linked variables.

(2) This brings the user to the stage of specifying a conditional probability table for each variable V_i given each combination of its parent nodes $PA(V_i)$. In general, they may be considered parameters of the system that need to be either estimated from data or subjectively assessed. When this is

complete, the probabilistic model is specified fully.

(3) The system initialises the model from the stored probability tables in order to be in a state ready to receive evidence and to be able to calculate efficiently quantities of interest, such as the marginal probabilities.

(4) The user enters observed evidence as it arrives, possibly in batches. The system absorbs the evidence accordingly.

(5) Evidence is propagated and belief is revised.

(6) The results are presented to the user.

Forming a computational model based on the probabilistic reasoning will involve two distinct stages: knowledge acquisition and evidence propagation. Steps (1) and (2) can be referred to as the knowledge acquisition process, through which an expert or a user may supply information about some area of knowledge in the form of a Bayesian belief network and associated conditional probabilities. Steps (3), (4) and (5) are the propagation process, where step (3) is known as initialisation. The propagation will deal with any individual case relating to an area of knowledge which has been dealt with by the knowledge acquisition process. Step (6) is concerned with displaying probabilities. We want to stress that the system allows the user to proceed step by step. That is, they can construct a simple network, initialise it and store it. In a subsequent step, they can retrieve it. Eventually they can modify it, enter evidence, run evidence propagation and so on.

Next, we illustrate the design principles described and give an overview of the operations of the system. We will use the basic terminology of the SunView environment (such as windows, buttons and mouse) without a closer explanation of these terms.

3.3.1. Knowledge Acquisition

There are two tasks carried out in this part: *create a model* and *input probability*. Structure and conditional probability tables are constructed gradually via a simple, interactive graphical interface.

A node is created by clicking the left mouse button over any blank area of the drawing board. A box representing the node is drawn at that screen position and an index number (starting from 1) is automatically associated with the box. Each node is then associated with a menu. Pressing the right mouse button inside the box results in the menu that will be displayed over the box and gives the following options: *set up*, *move*, *information*, *erase* and *modify*. The user is able to assess/change/display some properties of the node. All these will allow the user to manipulate nodes easily. If *set up* is selected, a small window will pop up and prompt the user to provide basic information about this node. The basic information includes its name, its possibles states, its parents and children. To create a binary node, the user may simply accept the two default values *yes/no* provided by the system. When this is completed, directed connections between the node and its parents or children are displayed automatically on the drawing board. Therefore, the network can be constructed gradually. The option *information* will show all properties for this node and *modify* will allow the user to carry out some changes. Apart from these operations, the existing nodes can be edited, that is, *move* and *erase*. These will allow the user to manage the structure and produce a desirable layout.

When the creation of a structure is finished, a dependency matrix is produced for internal representation of the Bayesian belief network. Each row of the matrix will correspond to a node label and the entries on that row will correspond to the labels of the nodes caused by the node. Then a syntactic check is carried out for testing the overall consistency of the Bayesian belief network. Some advice will be given if some nodes are left unlinked and if there are any directed loops.

If the structure has been determined, the model can be saved at any point, for later continuation by reloading. Furthermore, the system is ready to quantify the dependence relationships of the structure. This is carried out by means of conditional probability tables. Probability assessments are required for each node, given its direct parents, for every possible combination of states of parents. The system automatically calculates parent nodes for each node in the graph and all possible combinations of their states. In the meantime the storage of current input probabilities of nodes are displayed graphically in the display window. Manipulation of a conditional probability table is accomplished by first indicating a node and then selecting a menu. The menu has two options: *input* and *show*. The option *input* enables the user to specify conditional probabilities for the node. A popup conditional probability table is displayed, which

shows the state of the node and the states of its parents, for example see Figure 3.5. The skeleton for the table is automatically generated, and the number is typed in. When the probability values are input and the user clicks *done*, a new conditional probability table pops up until all possible probabilities have been supplied. Having input all probabilities of a node given its parents, a slide is filled with respect to the node to indicate that it is complete in the displaying window, see nodes P and Q in Figure 3.5. The option *show* will display the conditional probability table in a popup window. Now the probabilistic model is complete after the conditional probabilities of each node have been assessed.

Internally each node is attached to a conditional probability table. The table is designed to be two-dimensional. An index system is used to store all the possible conditional probabilities of the node properly. Given a node and the certain combination of its parent nodes, an index is calculated and then the corresponding probabilities can be easily accessed from the table. For example, if a discrete node V_i with N states has M discrete parent nodes, namely $F^1, F^2, ..., F^M$, and F^t has S_t possible states, then there are $NS_1S_2\cdots S_M$ values to be stored. It is unnecessary to use an (M+1)-dimensional table for storing and manipulating these probabilities. For a particular conditional probability $P(V_{ik} \mid F^1{}_{j_1}, ..., F^M{}_{j_M})$, it can be accessed with the entry (k,r), where the suffix k is the index to identify the state of V_i and r is calculated as follows:

$$r = j_1 + S_1(j_2-1) + \cdots + S_1S_2\cdots S_{M-1}(j_n-1)\,.$$

In order to get all possible combinations of conditional probabilities for a particular node given its parent nodes, a special procedure *get_combinations* is designed. The procedure produces a combination *comb* each time and can be described as:

```
get_combinations(comb, no_of_parents, finished)
{find a combination}
BEGIN
    j=no_of_parents
    finished=FALSE
    WHILE (j>1) AND (comb[j]≤1) DO
        j=j-1
    IF j≥1
    THEN  comb[j]=comb[j]-1
    ELSE
        finished=TRUE
    FOR   i=j+1  TO  no_of_parents  DO
        IF  (comb[i]>0) AND (comb[i]≠maxvalue[i])
        THEN    comb[i]=maxvalue[i]
END
```

where *maxvalue* is an array storing the number of states of each node. When all the possible combinations have been found, *finished* is set to be TRUE. For a node having three binary parent nodes, the combinations produced by the above procedure will be: (2,2,2), (2,2,1), (2,1,2), (2,1,1), (1,2,2), (1,2,1), (1,1,2), (1,1,1).

3.3.2. Propagation Process

PRESS provides both exact and approximate algorithms. For each algorithm, there are two possibilities: *Initialisation* and *Evidence Propagation*. Initialisation will structure the domain knowledge before any dialogue with a user. If there are some observed nodes, the evidence will be discarded and all the probabilities will be initialised from the conditional probability tables. Evidence Propagation will work when new evidence is entered or when queries are posed.

3.3.2.1. Initialisation

Although this appears simple, it actually holds the key to success of the propagation process. What happens is that the state space of the model is factorised in order to allow for efficient computations.

Having constructed the structure and specified the conditional probabilities, topological and numerical representation changes to the model are necessary in order to perform probabilistic reasoning efficiently. The preprocessor will detect the structure and use different methods accordingly. If it is a singly connected structure, Pearl's algorithm can be applied directly. Otherwise, the L-S algorithm is used. A clique tree is constructed and will serve as a computational data structure for propagating evidence later on.

From conditional probability tables and the clique tree, local representations on the cliques and separators are obtained [28]. The probability of a particular node is calculated by summing relevant members of the clique which contains the node. All these have been carried out in the background.

3.3.2.2. Evidence Propagation

Clicking the right mouse button inside a box, the user can specify either certain evidence or uncertain evidence. If the evidence is certain, all possible states of the node are displayed as a menu. Having selected a proper state in the menu, the evidence is input, that is, the selected state of this node is observed. The system propagates this evidence in the structure in the background. The updated posterior probabilities are calculated and ready to be displayed.

Although there are two ways to deal with evidence for the particular structure, the message-passing idea is used in evidence propagation. Note that each node is a unit in Pearl's approach, but each clique of the triangulated graph is a unit in the L-S approach. In fact, each unit communicates with its neighbours in a tree structure. Three steps are carried out for each unit in message-passing propagation. The first step is to collect the message from its neighbour. The second step is that the probability distribution on each unit is updated using Bayes' rule. The final step is to pass the impact of evidence to others through the links. This process repeats until all the units are updated.

3.4. Evaluation of Different Reasoning Algorithms

The development of PRESS provides us with an opportunity to investigate different algorithms (exact and approximate). In this section, we study the performance of three algorithms discussed in chapter 2 on a Sun 4 timesharing processor running SunOS, a version of 4.1 BSD UNIX. A empirical comparison of the algorithms has been made using PRESS. We measure CPU time with the UNIX system call clock(), which returns the elapsed processor time in microseconds.

For illustration purposes, we consider a network with 20 nodes. First, a tree structure (with 19 edges) is generated using a random number generator. Then an additional directed edge, which does not cause any directed loop in the network, is added at random each time by visiting the nodes one by one to assure a

fairly equal distribution. A series of data sets has been produced for evaluation of different reasoning algorithms.

Pearl's algorithm and the L-S algorithm have been tested on the these data sets. The CPU time of initialisation and propagation (a single piece of evidence) have been measured and presented in Table 3.1.

Nodes: 20	Pearl's Algorithm		L-S' Algorithm	
Edges	Initialisation	Propagation	Initialisation	Propagation
19	290	20	510	30
20	510	60	520	30
21	600	160	610	40
22	1010	610	810	40
23	3670	3330	830	50
24	5050	4660	830	60
25	20130	17520	850	90
26	49170	43030	870	100
27	98480	87400	9270	310
28	101350	89980	22500	310
29	203530	188720	23060	540

Table 3.1. CPU Time in Microseconds

The differences between the two algorithms are significant. Pearl's algorithm is more suitable for tree structure. On the other hand, the L-S algorithm is flexible and works well when the graph is large and sparse. For a network with 20 nodes and 26 edges, Pearl's algorithm takes 49 seconds to initialise the model but the L-S algorithm takes only 0.87 second. Updating the network with Pearl's algorithm takes approximate 43 seconds for a single evidence. On the other hand, evidence propagation with the L-S algorithm is completed in 0.1 second for the same evidence. Characteristics of the network topology and the handling of loops determine this behaviour. The performance of Pearl's algorithm can be explained by the size of loop cut nodes. The high connectivity of the network, a problem for Pearl's algorithm, makes the cliques smaller in the case of the L-S algorithm. As there is only a small number of parents per node, the maximum clique size stays within reasonable limits. In addition, the experimental results also confirm Cooper's analysis [80], that is, the computational complexity of both algorithms is NP-hard. For example, both algorithms need 0.5 second to initialise the network with 20 edges, but 203 seconds for Pearl's algorithm and 23 seconds for the L-S algorithm to initialise the network with 29 edges.

Unlike the exact algorithms, the benefit of stochastic simulation algorithms is that computational effort is unaffected by dependencies in the Bayesian belief network as a simulation of an event, given the

states of its causes, does not depend on correlation between those causes. Table 3.2 shows the CPU time for the stochastic simulation algorithm on the same data sets used for exact algorithms. Table 3.3 gives the CPU time of conducting 10,000 simulations for four different runs. Each simulation run requires only $n+e$ steps, where n is the number of nodes and e is the number of edges in the graph. The experimental results presented in Table 3.2 and Table 3.3 confirm this analysis. For example, given the sample size as 5000, it takes about 20 seconds to process the network with 20 edges, 24 seconds to process the network with 29 edges. Thus the length of computation is mainly determined by the level of accuracy required, see Table 3.2 and Table 3.3.

	Stochastic Simulation				
Edges	100	500	1000	5000	10000
19	420	2030	4000	19830	39640
20	450	2070	4130	20540	40980
21	460	2120	4220	21190	41940
22	450	2160	4310	21490	42870
23	470	2240	4380	21910	43890
24	470	2260	4530	22470	44890
25	470	2310	4640	23070	46160
26	480	2360	4660	23390	46630
27	510	2400	4890	23840	47690
28	510	2480	4960	24040	48150
29	530	2500	4950	24800	49420

Table 3.2. Running Time (ms) on Different Sample Sizes

	Stochastic Simulation			
Edges	Run 1	Run 2	Run 3	Run 4
19	39730	39660	39840	39640
20	40980	40990	41030	40980
21	41910	41900	41910	41940
22	42700	42850	43020	42870
23	43800	43960	43840	43890
24	44800	44840	44820	44890
25	46240	46020	46180	46160
26	46740	46590	46790	46630
27	47690	47670	47760	47690
28	48160	48140	48090	48150
29	49530	49560	49550	49420

Table 3.3. Running Time (ms) of Different Runs (10,000 simulations)

3.5. An Example

The use of PRESS is best described by means of an example. The example given here was suggested by C.G.G. Aitken [97,104]. The example is artificial but it illustrates the basic concepts of the Bayesian belief network representation and the functions of PRESS.

A murder has been committed. There are two suspects, X and Y, who are associates and who say they met the victim, V, some time before the commission of the crime. If there were an eyewitness to this meeting it would be interesting to know from that witness if the meeting had been cordial or not and, in particular, if there had been a fight. Since X and Y are associates it is feasible that Y may pick up something from X and then deposit it at the scene of the crime. For example, fibres from a jacket of X's may be picked up by some garment of Y's and then be left at the crime scene by Y, thus incriminating X who may, in fact, be perfectly innocent. Such transfer from X to Y may take place if, for example, Y drives X's car frequently.

Suppose the suspect X committed the crime. Using Locard's Principle that every contact leaves a trace, he will leave something behind at the scene. It is also possible, through the secondary transfer argument above, that if Y commits the crime, he will leave something of X's behind. Let us assume that this "something" is fibres from a jacket belonging to X. There are obviously other factors which could or should be taken into account [105]. For example, if Y committed the crime he will leave something of his own by which he might be identified as well as something of X's. However, for simplicity of exposition, these are omitted.

These different aspects of the investigation may be combined, graphically, to illustrate the relationships between them. There are different types of potential evidence which requires consideration. There is eyewitness evidence which may be unreliable. The idea that Y drives X's car frequently is important. It raises the possibility of secondary transfer. Suppose X wears his jacket while driving his car and leaves fibres on his car seat. Y then drives the car and picks up fibres from X's jacket on his own clothing. Y wears the same clothing when he commits the crime and leaves behind fibres from X's jacket at the crime scene.

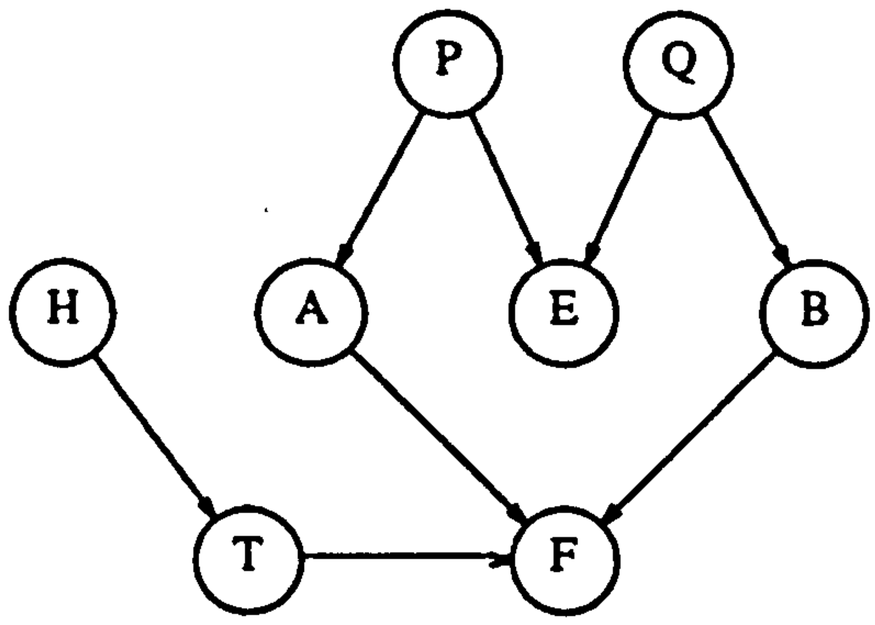


Figure 3.3. Bayesian Belief Network of the Forensic Science Example

The Bayesian belief network, displayed in Figure 3.3, summarises the relationships described in the example. A *node* in the graph represents a variable, which may be either a piece of evidence, for example that Y drives X's car frequently, or a conclusion, for example that the suspect X committed the crime. A list of the variables for our example is given in Table 3.4. We assume that each variable in our example may have two states, i.e. being *true* or *false*. We will use upper case letters to represent variables and lower case letters to represent their particular states, for example, t and ¬t (*true* and *false*) are two states of variable T. An arrow between two nodes, say P to E, represents a "causal" relationship and it is said that node P "precedes" node E in the graph.

Variable	Interpretation
P	X quarrels with V.
Q	Y quarrels with V.
E	Eyewitness evidence of a row between X,Y and the victim sometime before the commission of the crime.
A	X committed the murder.
B	Y committed the murder.
F	Fibres from jacket similar to X's are found at the crime scene.
H	Y drives X's car frequently.
T	Y picks up fibres from X's jacket from X's car.

Table 3.4. Variable List

The task is: given some assumptions of conditional independence reflected in the graphical structure, and prior probability distributions for nodes P, Q and H, and conditional probabilities of the other nodes in the graph given their parents, we need to calculate the updated probability distribution in the light of evidence. All these prior and conditional probabilities are given in Table 3.5. Eyewitness evidence of a row between X and V or between Y and V is unlikely. X and Y are unlikely to commit the crime *a priori*

but possibly might have, given eyewitness evidence. In addition, it is quite likely that Y drives X's car. If Y drives X's car frequently, it is possible that Y will pick up something from X and then deposit it. Fibres from X's jacket are quite likely to be found if X committed the crime, regardless of whether Y was involved or not. If Y did not pick up fibres from X's jacket and X did not commit the crime, it is unlikely that fibres from X's jacket will be found at the scene.

Suppose that the fibres evidence is accepted, that is, fibres found at the scene of the crime are found to be similar to those from a jacket found in the possession of X. Then we may wish to assess how that evidence changes our belief in the guilt or otherwise of X and Y.

P(p)=0.05	P(q)=0.05	
		P(f|a, b, t)=0.80
P(t|h)=0.20	P(e|p, q)=0.95	P(f|a, b, ¬t)=0.70
P(t|¬h)=0.01	P(e|p, ¬q)=0.65	P(f|a, ¬b, t)=0.70
	P(e|¬p, q)=0.65	P(f|a, ¬b, ¬t)=0.70
P(h)=0.70	P(e|¬p, ¬q)=0.05	P(f|¬a, b, t)=0.20
		P(f|¬a, b, ¬t)=0.03
P(b|q)=0.50	P(a|p)=0.50	P(f|¬a, ¬b, t)=0.01
P(b|¬q)=0.01	P(a|¬p)=0.01	P(f|¬a, ¬b, ¬t)=0.01

Table 3.5. Prior and Conditional Probabilities Table

The joint probability distribution of all variables P(P,Q,H,T,A,B,E,F) in our example can always be expressed as the product of conditional probabilities:

$$P(P,Q,H,T,A,B,E,F)$$
$$= P(P)P(Q|P)P(B|P,Q)P(E|P,Q,B)P(A|P,Q,B,E)$$
$$P(H|P,Q,B,E,A)P(T|P,Q,B,E,A,H)P(F|P,Q,H,T,A,B,E). \quad (3.1)$$

There are some conditional independence assumptions reflected in the graph, namely that the probability of a particular event is independent of all those events not adjacent to it, conditional dependent only on those events which are adjacent to it in the graph. For example, if the suspect X committed the crime, the knowledge concerning X quarrelling with the victim V provides no information about the likelihood of finding the similar fibres from X's jacket to the fibres found at the crime scene. Using these conditional independence assumptions, the joint distribution expression (3.1) can be simplified as [28,75]:

$$P(P,Q,H,T,A,B,E,F) = P(P)P(Q)P(H)P(T|H)P(A|P)$$

$$P(B|Q)P(E|P,Q)P(F|A,B,T).$$

3.6. Results and Discussion

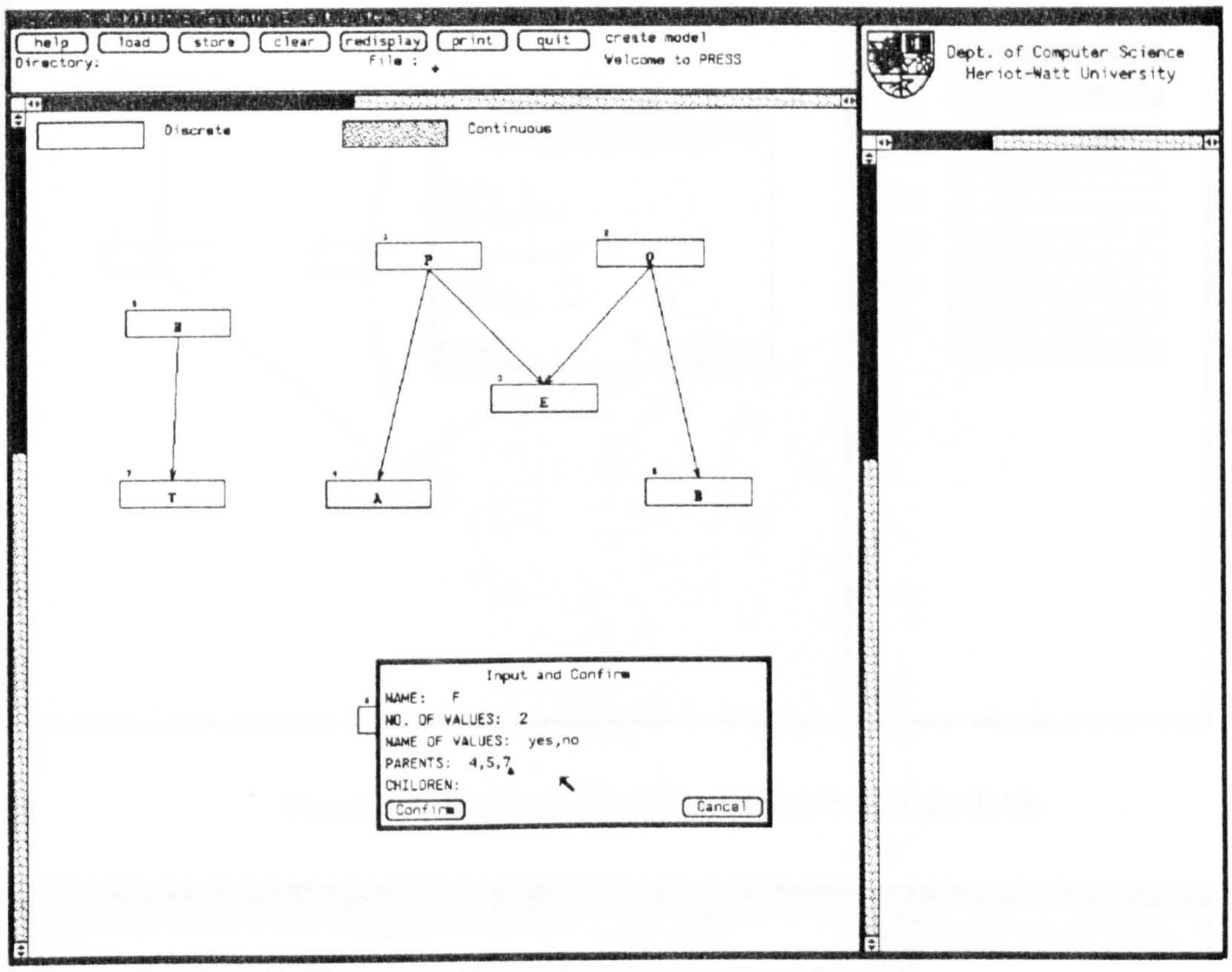


Figure 3.4. Creation of a Node

As we have described, our example is set up by using *create tool*. A node is created where desired in the graph by clicking the mouse over any blank area of the drawing board. A window then pops up over the drawing board for specifying the node, see Figure 3.4, where information about node F is provided. After the construction of the Bayesian belief network, conditional probability tables are used for representing quantitative knowledge. Figure 3.5 shows that conditional probability of node A, given its parent P as input. At that time, probabilities of P and Q are available. The resulting graphical structure for our example is shown in Figure 3.6.

The graphical structure of our example is not a singly connected structure, so the L-S algorithm is selected. A clique tree is constructed in initialisation procedure, shown in Figure 3.7.

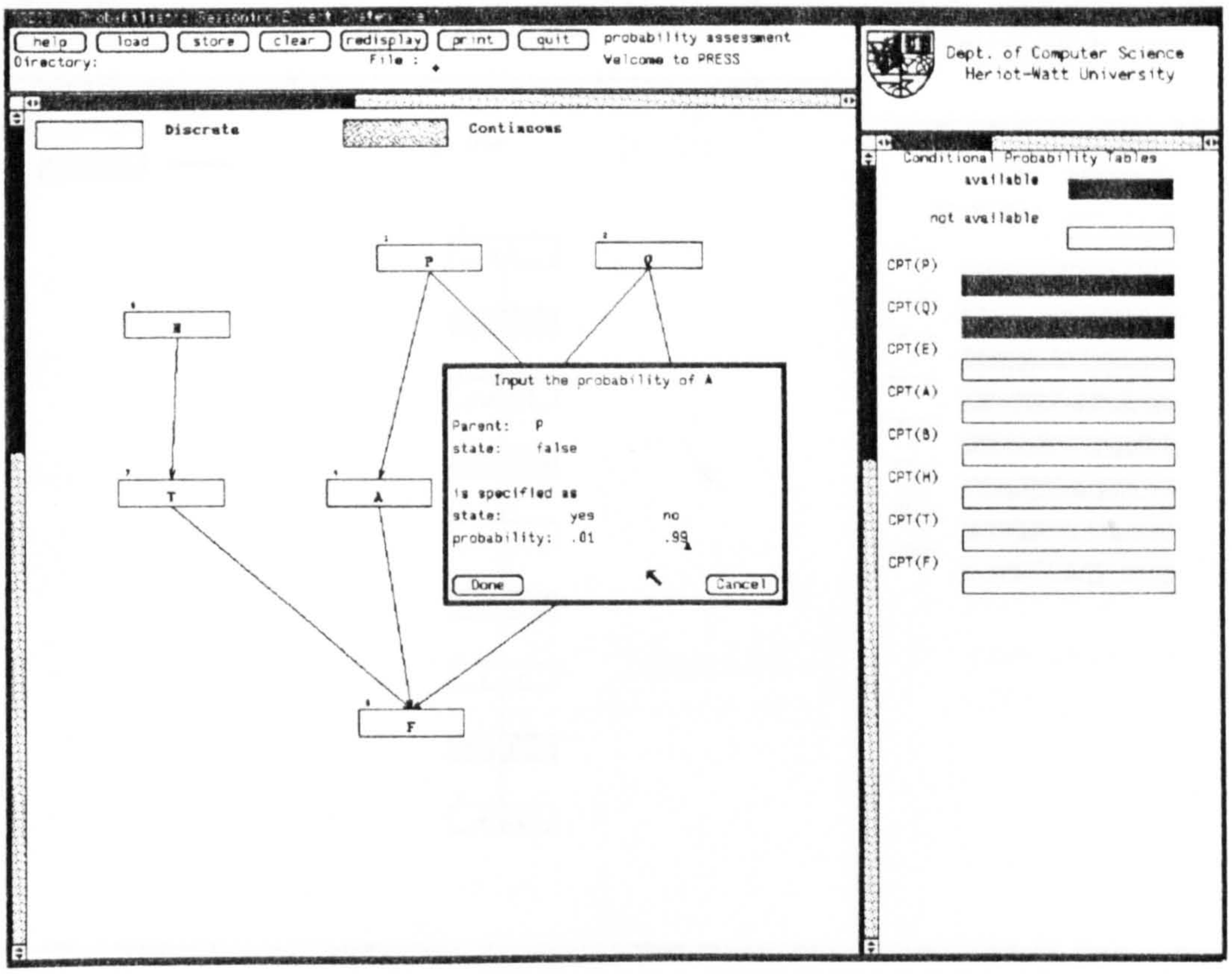


Figure 3.5. Manipulation of Conditional Probability Table

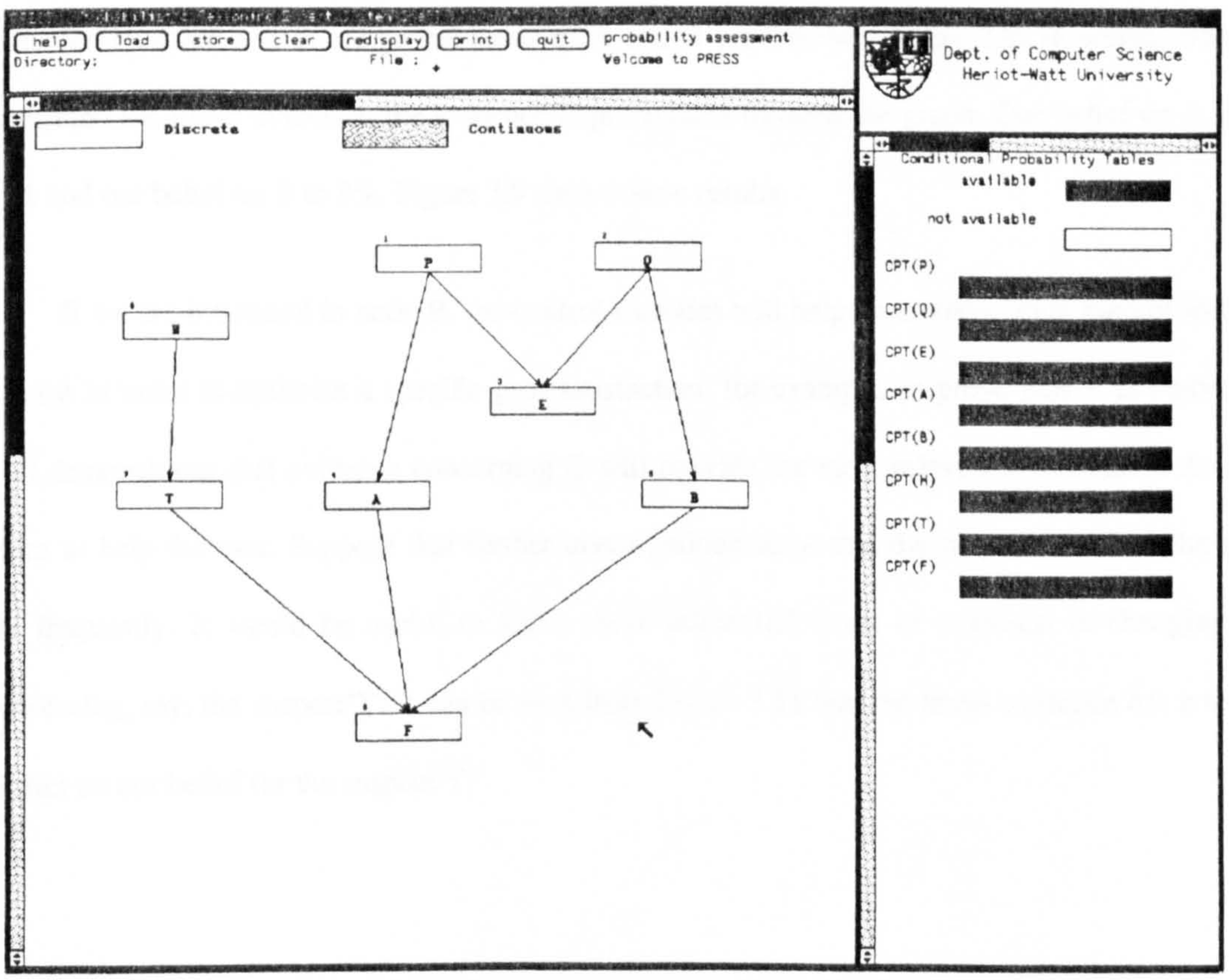


Figure 3.6. The Graphical Structure of the Example

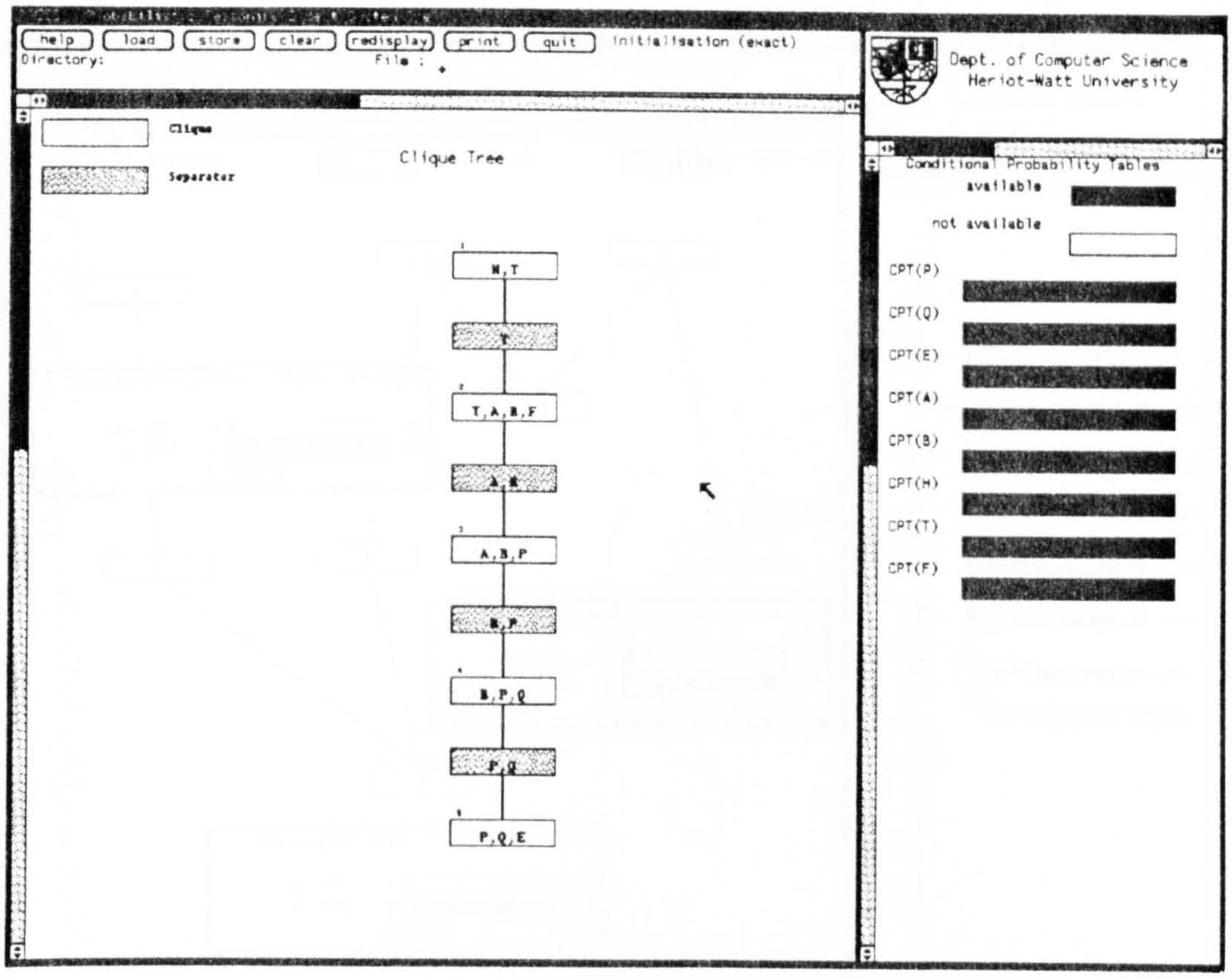


Figure 3.7. A Clique Tree of the Example

After initialisation, the initial probabilities of A and B are 3%, see Figure 3.8. However, when we have confirmed the fibres evidence, the evidence is propagated through the graph. Our belief on A increases to 68% and our belief on B to 8%. Figure 3.9 shows these results.

If we are interested in node B, the control facilities will help us to know what kind of information to request in order to optimise a specific goal satisfaction, for example, to prove that Y is innocent. Figure 3.10 demonstrates that evidence concerning Q will provide the most relevant information. An ordering is given to help the user. Suppose that further investigations show that the suspect Y drives the suspect X's car frequently. It would be useful to know these influential items of evidence in changing our belief concerning, say, the suspect Y. It can be seen from Figure 3.11 that the fibres evidence has a much greater impact on our belief for the suspect Y.

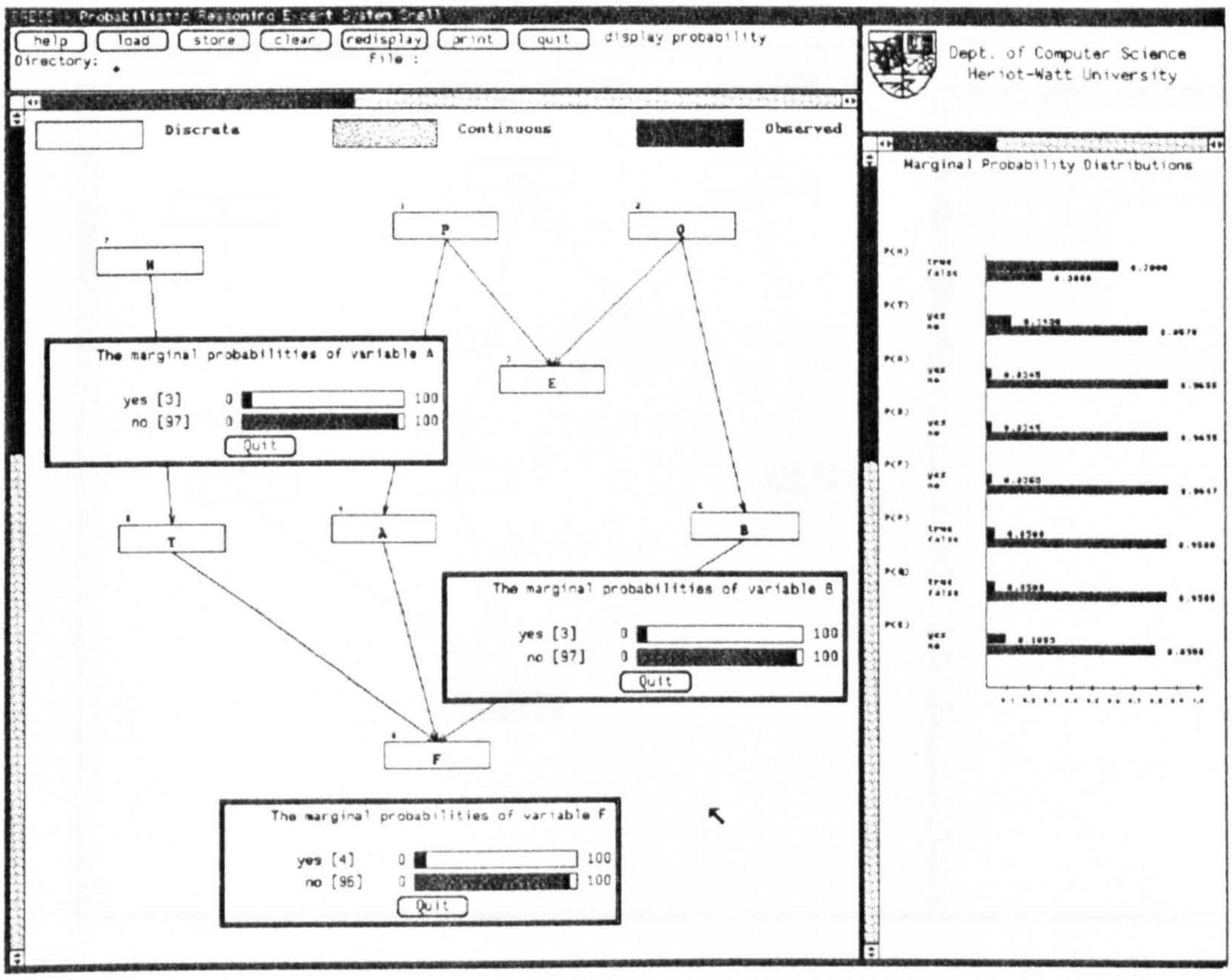


Figure 3.8. Initial Belief (exact algorithm)

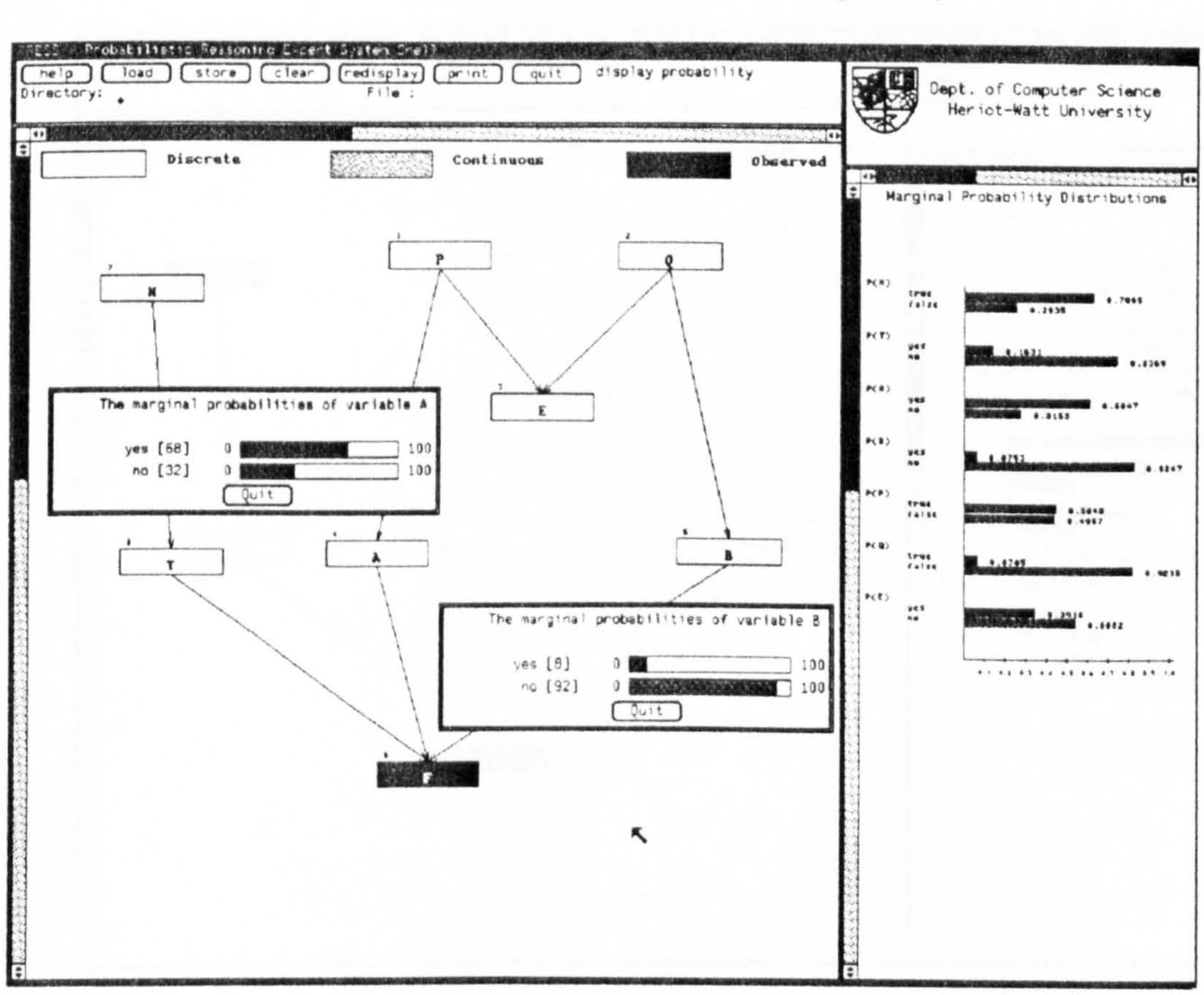


Figure 3.9. Updated Belief After Fibres Evidence (exact algorithm)

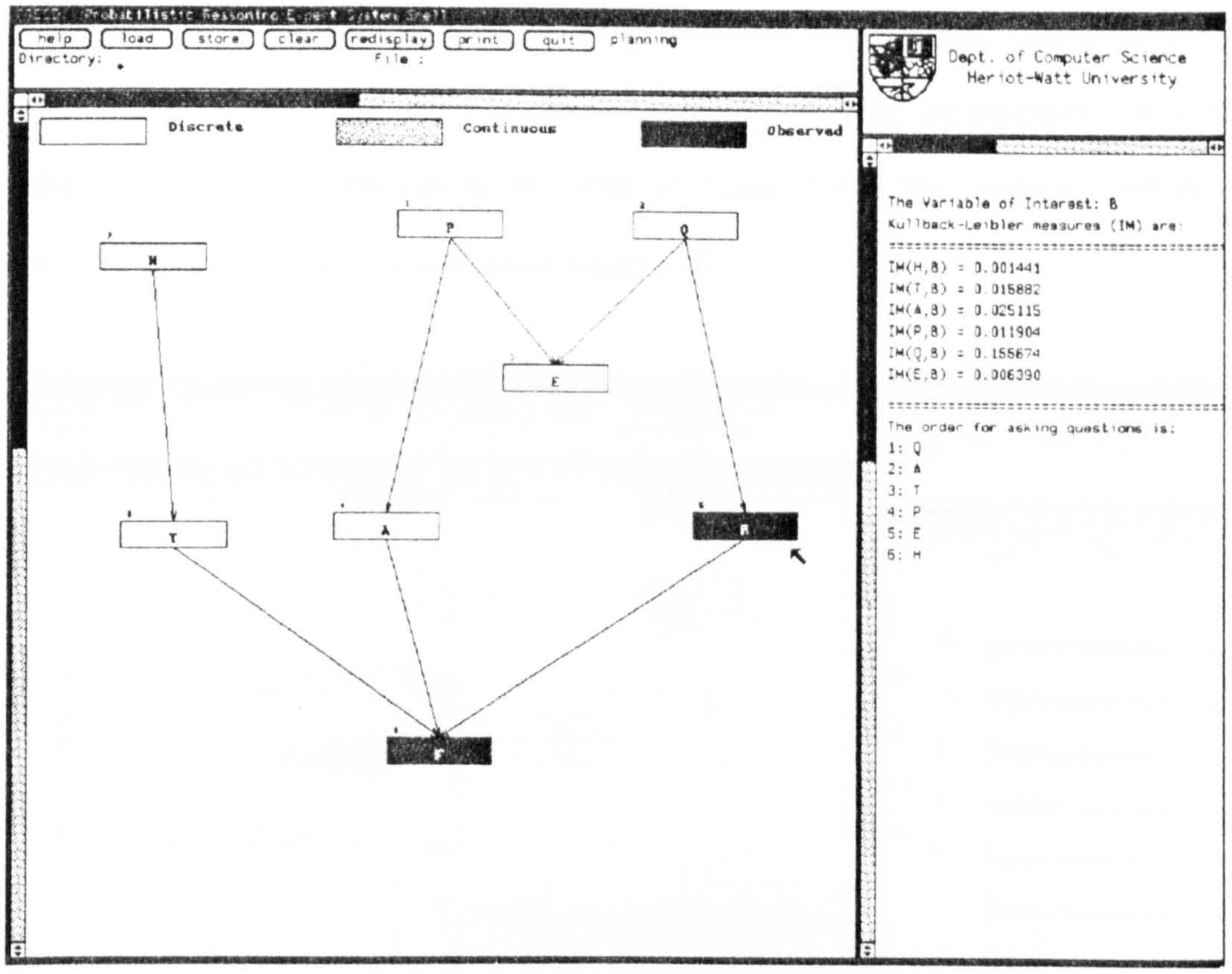


Figure 3.10. Planning

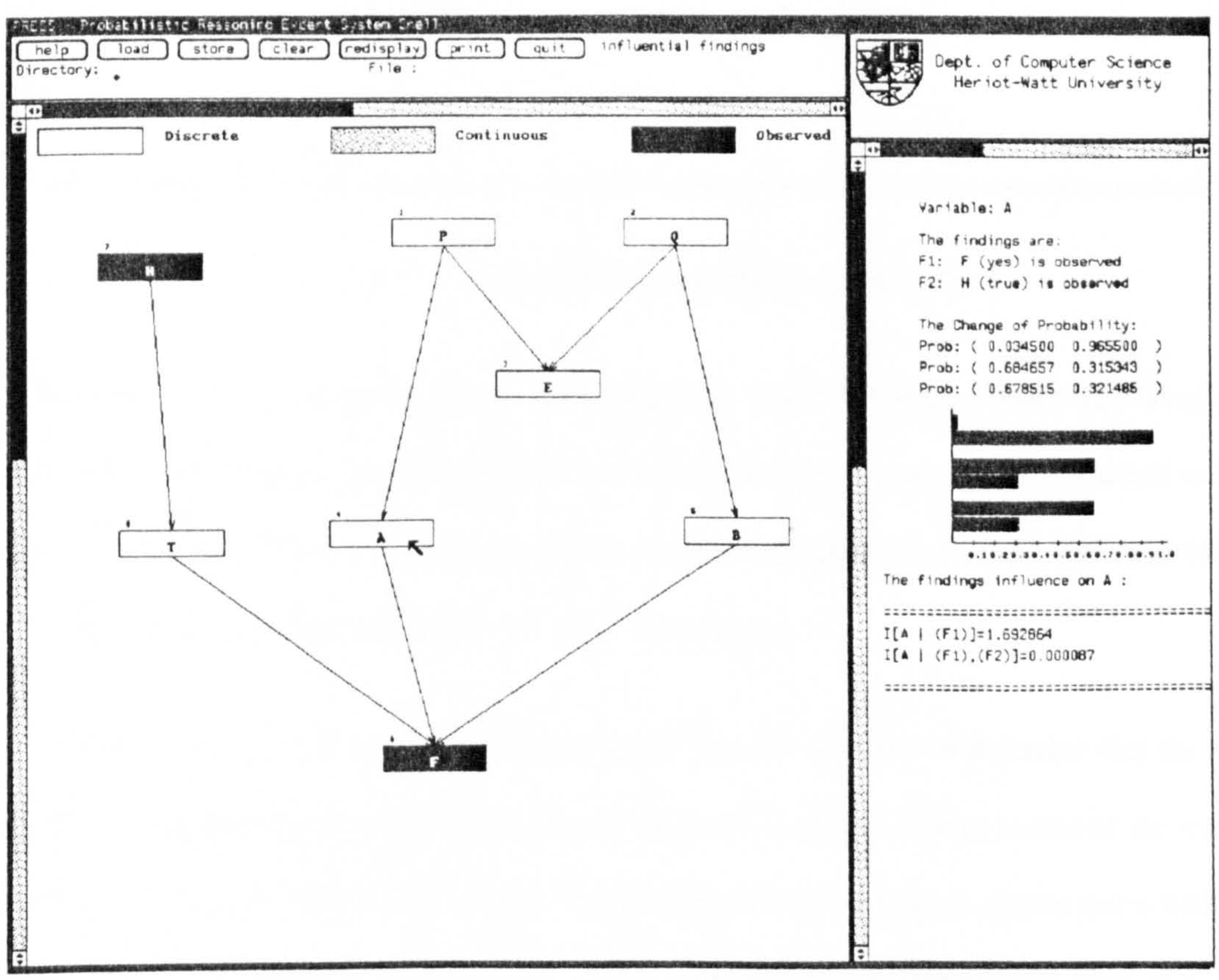


Figure 3.11. Influential Findings

The stochastic simulation method has been applied to the same example. The results are illustrated by Figure 3.12 and Figure 3.13. For demonstration purposes, we choose the simulation size as 1000. The estimated initial marginal probabilities are given in Figure 3.12. The estimated probabilities after acceptance of the fibres evidence are shown in Figure 3.13.

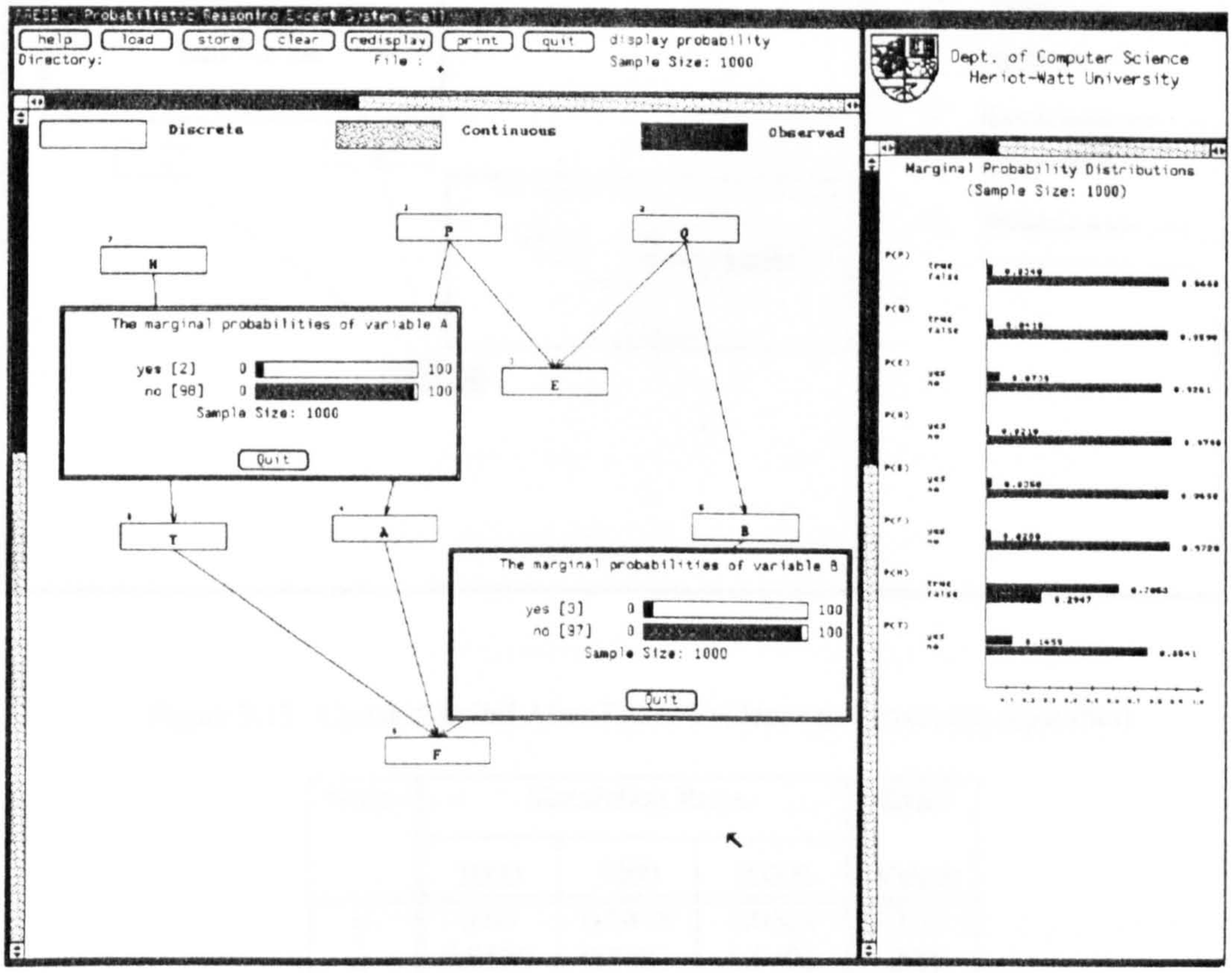


Figure 3.12. Initial Belief (approximate algorithm)

It is interesting to compare the results of the stochastic simulation and the exact algorithms using the same example. The marginal probabilities from the simulation, together with those calculated in the exact algorithm are shown in Table 3.6. Here we perform stochastic simulation with 1000, 5000 and 10000 runs. Table 3.7 shows the stochastic simulation with observed evidence.

It is clear that the results of the stochastic simulation are "optimal" in the sense that the estimated probability of a proposition using the stochastic simulation is "relatively" close to that of the exact value. For example, the prior probability of A, P(a) in Table 3.6 is 0.0345 in the exact system and is 0.0335 in the stochastic simulation where the sample size is 10000. Moreover, they remain close after an evidence is propagated. For example, when the "fibres evidence" is observed, our belief in A is 0.6847 using the exact

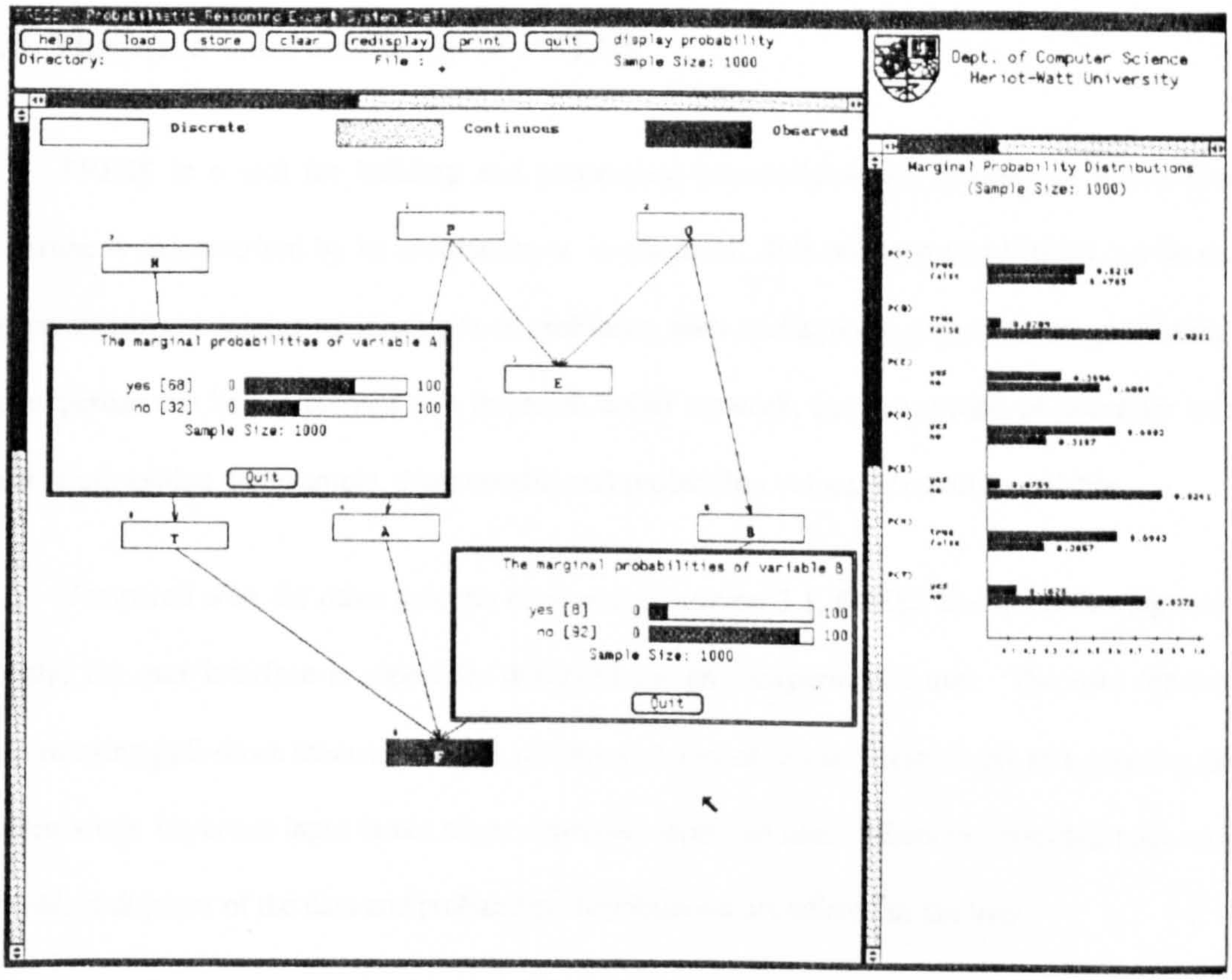


Figure 3.13. Updated Belief After Fibres Evidence (approximate algorithm)

Node	Simulation Runs			Exact
	1000	5000	10000	Values
p	0.05	0.0478	0.0502	0.05
q	0.0599	0.0494	0.0496	0.05
e	0.0509	0.0512	0.0495	0.05
a	0.03	0.0300	0.0335	0.0345
b	0.047	0.0330	0.0358	0.0345
h	0.6853	0.7036	0.6965	0.7
t	0.1149	0.1512	0.1415	0.1430
f	0.03	0.0320	0.03400	0.0353

Table 3.6. Simulation Results (Initial) for the Example

Node	Simulation Runs			Exact
	1000	5000	10000	Values
p	0.5355	0.4955	0.4935	0.5043
q	0.0769	0.0804	0.0775	0.0785
e	0.0549	0.0476	0.0491	0.05
a	0.7003	0.6827	0.6733	0.6847
b	0.0699	0.0762	0.0750	0.0753
h	0.7063	0.7099	0.7067	0.7065
t	0.1499	0.1616	0.1618	0.1631

Table 3.7. Simulation Results (Updated) for the Example

algorithm and 0.6733 using the stochastic simulation algorithm, see Table 3.7.

3.7. The Main Characteristics of PRESS

PRESS is a tool for building and processing knowledge-based systems in which knowledge or expertise is characterised by its uncertainty or inexactness. It is believed that PRESS can be used to build expert systems in solving various types of problems, such as diagnosis, classification, prediction, as long as the expertise can be represented as a Bayesian belief network, and acquisition of uncertain knowledge on their relationships, for example, those conditional probability values, is readily available.

Compared with the other systems reviewed in section 3.1, PRESS has the following characteristics. Firstly, the user interface is especially designed for an inexperienced user. The user interacts with the system using pull-down menus, dialogue windows. A mouse is used extensively as a pointing and selection device while keyboard input is not often requested from the user. Whenever possible both *numerical* and *graphical displays* of the data and probability distributions are offered to the user.

Secondly, PRESS allows different strategies in evidence propagation and provides both exact and approximate algorithms. These two types of algorithms have been integrated in the same computational framework. Therefore, it can be applied to a wide range of problems.

Thirdly, PRESS offers control facilities: planning of effort or controlling of reasoning process. Moreover, a help facility is available for the user. By means of this feature the user obtains a brief description of each menu item. The clique tree, which gives some insight of the model, is also presented to the user when the exact algorithm is selected.

Fourthly, PRESS enables the user to proceed step by step. The user can experiment with different structures and different algorithms for evidence propagation.

Finally, the open architecture of PRESS enables us to study, develop and accommodate other related procedures in Bayesian belief networks. So PRESS can either work stand alone or generate knowledge sources for knowledge based systems.

As with any other uncertain inference systems, PRESS has some limitations and there are aspects which can be improved in the system. Construction of a graph could be a very tedious process, especially when large graphs are involved. An automatic graph layout technique is a possibility. Extracting graphical

knowledge representations from written text is another possibility. Natural language processing will be the key issue.

At the moment, PRESS can provide marginal probabilities of each variable and distributions on cliques and separators. However, sometime the user may want to know the distribution on a set of variables.

Multimedia techniques could be used to enhance the current user inference. The thickness and colour of the dependence arrows can be used to reflected the probabilities of the effects given their parents. These would provide a quick visual indication to the user of important areas of the structure on which to focus.

Expert systems should be able to explain their reasoning in terms comprehensible to their users. Most people express a preference for using natural language phrases. An explanation facility translating from numerical to natural language would be an advantage.

PRESS is designed for probabilistic reasoning in Bayesian belief networks but it could be extended to handle influence diagram and solve decision problems. In that case, the networks will incorporate additional variables representing the candidate decisions and the decision-maker preferences.

3.8. Concluding Remarks

In this chapter, a flexible probabilistic reasoning expert system shell, which is referred to as PRESS, for modelling the problems and performing probabilistic reasoning is described and implemented. Two types of probabilistic reasoning mechanisms, exact algorithms including Pearl's algorithm [25] and the L-S algorithm [28], and approximate algorithm (stochastic simulation), are integrated into this shell system. They are selected according to particular structures of problems.

The shell is designed and implemented in an object-oriented programming style and an interactive graphical way. PRESS has been evaluated on an artificial example in the field of forensic science. We believe that this shell system makes construction and manipulation of Bayesian belief networks much easier and provides an opportunity for the user to explore probabilistic reasoning fully. These achievements have been possible through the steps taken during the development of PRESS. A critical examination together

with careful evaluation on existing probabilistic approaches treating uncertainty in expert systems, enabled a domain independent shell system to be developed, namely PRESS.

More importantly, an open computational architecture is designed and developed, which is very flexible and allows us to develop other functional modules. At this stage, probabilistic reasoning algorithms for pure discrete variables are developed in PRESS. However, real world problems often contain many continuous quantities and mixed models with both discrete and continuous variables are often needed. The open architecture of PRESS makes it possible to incorporate the extension of the new algorithm that allows the use of a mixture of the discrete and continuous variables. Work has been carried out on this aspect [34,35] and will be described in the next chapter.

Chapter 4

PROBABILISTIC REASONING IN MIXED MODELS

Much work has directly involved graphical models in expert systems in the case of only one kind of variable, either for discrete variables [25,28], or for continuous variables [29,106]. However managing both continuous and discrete variables is a primary task of the latest research into expert systems design [58,107,108]. Variables such as measurements are naturally continuous. For example, Bellazzi et al [107] and Andreassen et al [108] describe Bayesian belief networks applications to monitoring problems. The drug-response sensitivity of a patient during therapy is represented as a variable, whose probability distribution is achieved by a learning process over some patient observations. Both the observations and their standard deviations are represented as continuous variables.

When continuous variables are involved, the algorithms implemented in PRESS have to cope with the problems of changing continuous variables into discrete variables by discretising their probability distributions. The discretisation is referred to as the process that produces the partition of intervals from the range of the continuous random variable. For example, temperature may be partitioned into three intervals corresponding to the states: *high*, *medium* and *low*. However, a compile-time discretisation (before propagation) may be hampered by the difficulty of deciding on the most appropriate partition of intervals. A run-time ("dynamic") discretisation carries a significant overhead, particularly when complex distributions are involved. In order to obtain a sufficient precision a large number of states are often required. This results in huge demands on space, which, in turn, gives long response times.

To address this problem, a computational procedure of probabilistic reasoning in mixed models is described and implemented in this chapter [35]. This procedure is based on PRESS [33] due to its open architecture. The models are mixed in the sense that some of variables are continuous with a special class of Conditional Gaussian (CG) distribution, and some are discrete [109,110]. The theory behind this approach has been developed recently by S. Lauritzen [34], and is based on directed Markov fields of CG-

distributions [111].

The contents of this chapter are arranged as follows. Model specifications for mixed continuous and discrete variables are described in section 4.1. In section 4.2 internal representations are investigated. Section 4.3 specifies the operations on the internal representations. Modifications of PRESS are designed and implemented in section 4.4. Section 4.5 gives an example of the forecast of crop yield to illustrate the extended PRESS [35]. Finally, the approach is discussed in section 4.6.

4.1. Model Representation

The models considered in this chapter are those whose dependency structure can be modelled graphically [28,112]. The dependency structure where nodes represent variables and directed edges represent associations between linked variables, is given by, for example, the domain experts. However, the difference between the models in this chapter and those in the previous chapter is that variables can be either discrete or continuous. Discrete variables are described by a finite number of states. Each node is in exactly one of its states, but knowledge about which state will often be incomplete. On the other hand, continuous variables are not restricted to a finite number of states, but can take on (at least in principle) any real value. The continuous variables are assumed to have a special class of probability distributions, namely CG-distributions, see Appendix C. The structure reflects the fact that two variables which are separated by a third are conditionally independent given that the value of the third is known [28]. When a mixture of both discrete and continuous variables is considered, a restriction is introduced: continuous nodes are only allowed to have continuous children [34].

To complete the model, a numeric description of the relations has to be added. This is carried out in terms of conditional probability tables. For discrete nodes, the probability of each of its states occurring is specified given all combinations of states its parents could take. Thus a table can be thought of as entries, one for each parent configuration, holding the distribution over the states in the node. If no parent node is present, the table reduces to unconditional probability. This is exactly the same as that in the pure discrete case.

On the other hand, it is assumed that all continuous variables are normally distributed, perhaps after a transformation, and that they combine linearly. Different situations are distinguished by the following:

(i) A continuous variable which is not dependent on other variables requires specification of the mean and variance;

(ii) A continuous variable which is dependent on discrete variables requires specification of the mean and variance for each possible value, or combination of values, of the discrete variables on which it depends;

(iii) A continuous variable which is dependent on other continuous variables requires specification of its mean as a linear function of the values taken by the other variables and of its variance as a constant, independent of the values of the other variables;

(iv) A continuous variable which is dependent on both continuous and discrete variables requires specification of its mean as the sum of a value determined by the values of the discrete variables and of a linear function of the values taken by the continuous variables. The variance is dependent on the values taken by the discrete variables only.

As for the discrete case it is assumed that the joint density of the variables is the product of the conditional densities of the variables attached to each node, given the states of their parent nodes. In the mixed case, an assumption is made that the joint distribution of mixed collections of variables is a CG-distribution [109]. In general, depending on the values taken by its parent nodes $\mathrm{PA}(V_i)$, a conditional probability $P(V_i \mid PA(V_i))$ is substituted by $F(V_i \mid PA(V_i))$, the conditional density distribution of V_i in which the parameters are functions of the parents values. The conditional density of a continuous variable C given its parents (for example, discrete parents I and continuous parents Y) has the following format

$$F(C \mid PA(C)) = \mathrm{N}(\alpha(i)+\beta(i)^T Y, \gamma(i)), \tag{4.1}$$

where $\alpha(i)$ is a real number, $\beta(i)$ is a vector specifying a linear function of continuous Y in the set of parent nodes PA(C), $\gamma(i)$, a positive real number, represents the variance, which is assumed to be independent of the values of the continuous variables and v^T denotes the transpose of the vector v. $\alpha(i)$, $\beta(i)^T$ and $\gamma(i)$ may be dependent on the states of discrete parent nodes. Thus for each configuration of discrete parents i a distribution $\mathrm{N}(\alpha(i)+\beta(i)^T Y, \gamma(i))$ is specified. By analogy with the discrete case, the result is a table of

parameters of the distributions, with an entry for each possible combination of discrete parent nodes.

4.2. Internal Representations

Having specified the structure and conditional probability tables, we need to be able to calculate the marginal mean and variance of every continuous variable and the marginal probability of every discrete variable. However, the original structure and the conditional probability tables are not appropriate for such calculations. A static secondary structure used for computation is created. The idea is to decompose the structure into cliques so that computations can be performed locally on them [28,111]. A strongly decomposable graph is made [113] and a tree of cliques, which will serve as a computational data structure for evidence propagation, is built. Each clique in the decomposable graph forms a node in the cliques tree, and the separator sets (the intersections of neighbours in the clique tree) form edges [30].

Next, we consider quantitative representations on the clique tree. These internal representations are called CG potentials [109]. The notation of a CG-distribution may be extended to that of a CG potential $\phi(x)$

$$\phi(x) = \phi(i, y) = \chi(i)\, exp\left(g(i) + h(i)^T y - \frac{1}{2} y^T K(i) y \right),$$

where the only assumption is that $K(i)$ is a symmetric matrix. All information stored on the model is represented as CG potentials and the algorithms for evidence propagation depend on efficient manipulation of these potentials. CG potentials can still be represented by either the canonical characteristics triple $(g(i), h(i), K(i))$ or the moment characteristics triple $\{p(i), \xi(i), \Sigma(i)\}$. The moment characteristics are only well defined when $K(i)$ is positive definite for all i with $\chi(i)>0$. Then $\xi(i)$ and $\Sigma(i)$ are calculated as below. We can calculate one from the other easily using the following relationships (see Appendix C):

$$K(i) = \Sigma(i)^{-1}, \qquad h(i) = \Sigma(i)^{-1}\xi(i),$$

$$g(i) = \log p(i) + \frac{1}{2}\left\{ log\left(det\, \Sigma(i)^{-1}\right) - |\Gamma| log(2\pi) - \xi(i)^T \Sigma(i)^{-1}\xi(i) \right\}, \tag{4.2}$$

and

$$\Sigma(i) = K(i)^{-1}, \qquad \xi(i) = K(i)^{-1} h(i),$$

$$p(i) = (2\pi)^{-\frac{|\Gamma|}{2}} (detK(i))^{-\frac{1}{2}} exp(g(i) + \frac{1}{2}h(i)^T K(i)^{-1} h(i)), \tag{4.3}$$

where W^{-1} is the inverse matrix of W, $|\Gamma|$ is the number of continuous variables in the CG potential above, $g(i)$ and $p(i)$ are real numbers, $h(i)$ and $\xi(i)$ are vectors, and $K(i)$ and $\Sigma(i)$ are symmetric matrices. In general, CG potentials can be stored as tuples of a real number, a vector and a matrix. However, if $\Gamma=\emptyset$, only the real number is needed, whereas it is superfluous if $\Delta=\emptyset$.

4.3. Operations on CG Potentials

To perform reasoning within the mixed models, we need to be able to manipulate CG potentials with ease. We now investigate some basic operations on CG potentials. They include: extension, multiplication, division and marginalisation. Marginalisation is the most important and complicated operation with respect to our computational procedure since the adding of two CG potentials typically will result in a function of a different structure. Marginalisation can be further divided into three different cases, namely, marginal over continuous variables, marginal over discrete variables and marginal over both discrete and continuous variables. For details, see Appendix C.

Extension

Given a CG potential $\phi(y,i)=(g(i), h(i), K(i))$ defined on the space $I \times Y$, the extended CG potential $\phi'(y,z,i,j)=(g'(i,j), h'(i,j), K'(i,j))$ on the space $I \times J \times Y \times Z$ is defined as

$$g'(i,j) = g(i), \quad h'(i,j) = \begin{bmatrix} h(i) \\ 0 \end{bmatrix}, \quad K'(i,j) = \begin{bmatrix} K(i) & 0 \\ 0 & 0 \end{bmatrix},$$

where $I, J \in \Delta$ and $Y, Z \in \Gamma$.

Multiplication

Suppose there are two CG potentials $\phi_1(y,i)$ and $\phi_2(y,i)$ which are represented by canonical characteristics $(g_1(i), h_1(i), K_1(i))$ and $(g_2(i), h_2(i), K_2(i))$ respectively. If they are defined on the same space $I \times Y$, the product of these two potentials is calculated as:

$$\phi_1(y,i) \times \phi_2(y,i) = (g_1(i), h_1(i), K_1(i)) \times (g_2(i), h_2(i), K_2(i))$$

$$= (g_1(i) + g_2(i), h_1(i) + h_2(i), K_1(i) + K_2(i)) .$$

If they are not defined on the same space, we calculate an extension first and then carry out multiplication on the extended space.

Division

However, in the case of division special care has to be taken when dividing by zero. Consider two CG potentials $\phi_1(y,i)$ and $\phi_2(y,i)$ represented by $(g_1(i), h_1(i), K_1(i))$ and $(g_2(i), h_2(i), K_2(i))$ respectively. If $\phi_1(y,i)=0$, then $\phi_1(y,i)+\phi_2(y,i)=0$. If $\phi_2(y,i) \neq 0$, the division of these two potentials is defined as:

$$\phi_1(y,i)+\phi_2(y,i) = (g_1(i), h_1(i), K_1(i))+(g_2(i), h_2(i), K_2(i))$$

$$= (g_1(i) - g_2(i), h_1(i) - h_2(i), K_1(i) - K_2(i)) .$$

Marginal Over Continuous Variables

Assume that we marginalise a CG potential $(g(i), h(i), K(i))$ defined on lm+nl dimensions. The set of random continuous variables is partitioned into two parts: m and n, where the set m corresponds to those continuous variables we are going to marginalise. So $h(i)$ and $K(i)$ can be represented as:

$$h(i) = \begin{bmatrix} h_1(i) \\ h_2(i) \end{bmatrix}, \qquad K(i) = \begin{bmatrix} K_{11}(i) & K_{12}(i) \\ K_{21}(i) & K_{22}(i) \end{bmatrix},$$

where h_1(i) is defined on m and h_2(i) is defined on n. The marginalisation is carried out by integration. If K_{11}(i) is positive definite, the marginal potential with canonical characteristics $(g^*(i), h^*(i), K^*(i))$ defined on the dimension lnl is:

$$g^*(i) = g(i) + \frac{1}{2}\{m \log(2\pi) - \log (\det K_{11}(i)) + h_1(i)^T K_{11}(i)^{-1} h_1(i)\},$$

$$h^*(i) = h_2(i) - K_{21}(i) K_{11}(i)^{-1} h_1(i), \qquad K^*(i) = K_{22}(i) - K_{21}(i) K_{11}(i)^{-1} K_{12}(i).$$

Marginal Over Discrete Variables

Marginalisation has the complication that the simple adding of two CG potentials will result in a function of a different structure. Since we are dealing with the exponential of a component sum, adding two such exponentials will not result in a function similar to the original two. The essential difference between the pure discrete case and the situation with both discrete and continuous nodes is due to this.

This apparently leaves marginalisation undefined, but the problem is resolved by introduction of so-called *weak marginalisation*, operating on the moment characteristics. The idea is to approximate the marginal distribution by a CG potential whose moment characteristics agree with the true marginal moments. Suppose we have a CG potential $\{p(i,j), \xi(i,j), \Sigma(i,j)\}$ on discrete variables space $I \times J$ and we are going to sum over the discrete space J. If $K(i,j)$ is positive definite, the marginal potential will have $(p^*(i), \xi^*(i), \Sigma^*(i))$, where $p^*(i), \xi^*(i)$ and $\Sigma^*(i)$ are calculated as follows:

$$p^*(i)=\sum_j p(i,j), \qquad \xi^*(i)=\sum_j \xi(i,j)p(i,j)/p^*(i),$$

$$\Sigma^*(i)=\sum_j \Sigma(i,j)p(i,j)/p^*(i) + \sum_j (\xi(i,j)-\xi^*(i))^T(\xi(i,j)-\xi^*(i))p(i,j)/p^*(i).$$

This calculation ensures that the result has the correct expectation $\xi^*(i)$ and variance $\Sigma^*(i)$ [34]. Though only these two quantities are extracted it should be stressed that the complete information to construct the joint distribution is still represented internally.

Marginal Over Both Continuous and Discrete Variables

When we deal with this kind of marginalisation, we first marginalise over the continuous variables and then over the discrete variables. It is a combination of the above two methods.

Shift between Representation Types

In order to be able to shift between the moment and the canonical characteristic representations two switch operations are provided, see equations (4.2) and (4.3) in section 4.2. These operations involve various linear algebra functions, including the computation of determinants and matrix inversions. Special care has to be taken with respect to matrix inversion, as this operation is quite sensitive to computational accuracy.

4.4. Computational Design and Implementation Inside PRESS

The computational design of probabilistic reasoning in mixed models is based on the open architecture of PRESS. It will use the main parts of PRESS: Model Construction, Preprocessor, Enter Evidence and Evidence Propagation. The dependency structure and conditional probabilities are set up in

the Model Construction procedure. When the model is completed, a computational structure and internal representations for evidence propagation are built up automatically and the initial marginal for each node is calculated in the Preprocessor. When evidence is observed, it is presented to the system by the Enter Evidence procedure and absorbed. Finally, all the evidence is propagated in the Evidence Propagation procedure, and updated marginal probabilities, and means and variances are calculated.

The following describes the modifications required to deal with a mixture of continuous and discrete variables in PRESS. The user need not worry about these, seen from their point of view all that is needed is a specification of a model.

4.4.1. Model Construction

In this procedure, both qualitative and quantitative knowledge can be provided by the domain experts through the interactive interface. Qualitative knowledge is represented as a directed acyclic graph. The nodes of the graph are random variables which are either continuous or discrete and direct influences of a node V_i are "parents" of V_i in the graph. The variables are named. If a variable is discrete, its values are specified. A continuous node is distinguished graphically from a discrete one by a shadowed box. -2 is assigned to a continuous variable for internal representations. The relations between variables are then input. A dependency matrix is then set up, based on these relations. Note that continuous nodes can not have any discrete children. Quantitative conditional probability distributions for each node, given each configuration of its parent nodes, are needed based on the dependency matrix. For continuous variables, the mean and variance of the CG-distributions are specified. Conditional probability tables are then attached to each node for the Preprocessor.

4.4.2. Preprocessor

The clique tree representation is derived from the original dependence structure but with a specific property in the Preprocessor. The process is analogous to the discrete case except for one point. For pure models, the associated clique tree has the advantage that any node can be used as a root for propagation. Recall that the weak marginalisation only operates on the moment characteristics of a CG potential. In

general the matrix K of the canonical characteristic is not invertible, thereby preventing transformation of the potential to its moment characteristic. This obstacle disappears, however, if it is ensured that the initial traversal of the clique tree avoids weak marginalisations. This is obtained by a so-called **strongly** decomposable graph with a **strong root**, due to the asymmetry between discrete and continuous variables [34]. This implies that the graph is triangulated and does not contain any forbidden paths. A forbidden path is a path between two non-adjacent discrete nodes passing only through continuous nodes.

If G_{SD} is strongly decomposable, then G_{SD} can be successively be decomposed into simple models corresponding to complete subgraphs. In the same way as for triangulated graphs, strongly decomposable graphs can be characterised by the existence of certain orderings of nodes. The orders we are interested in are called "reducible ordering" such that the nodes in Δ are ordered lower than those in Γ. If $C_1, ..., C_n$ is a sequence of sets, typically the cliques of a graph, then for all $i \in [1,n]$, $S_i = C_i \cap (C_1 \cup ... \cup C_{i-1})$ and $R_i = C_i - S_i$. The sequence is said to be SD-ordered (strong decomposition ordered) [113] if furthermore for every $i \in [2,n]$

$$S_i \subseteq \Delta \quad or \quad R_i \subseteq \Gamma . \tag{4.4}$$

A strongly decomposable graph is one that can be reduced to cliques by strong decompositions. A strong decomposition of a graph is an undirected graph with the property that if $V = A \cup B \cup C$ (where A, B and C are disjoint subsets of the node set V in the graph) and the three conditions below are all satisfied [113,110]:

(1) C separates A from B;

(2) C is a complete subset of V;

(3) C is discrete or B is continuous.

An algorithm for moving from triangulated graphs to strongly decomposable graphs is given [113]. The algorithm consists of two basic steps:

(1) for any $k \in N$, generate an ordering α of the nodes in G

(2) test whether α is strongly reducible for G.

A graph is strongly decomposable if and only if its cliques can be arranged as a clique tree with a strong root. So if any two cliques C_i and C_j have common nodes, then those nodes are contained in all cliques in the unique path between C_i and C_j. A clique C_R on such a tree is a strong root if any cliques C_i, C_j are neighbours on the tree with C_i closer to C_R so that C_j holds: nodes in R are continuous and S are discrete. In general, the procedure of decomposing the structure is similar to one described earlier in chapter 2. Essentially it consists of the following steps:

(1) make the graph triangulated

(a) marry parents, that is, add a link between unjoined parents of a common child

(b) drop directions

(c) carry out a maximal cardinality search

Number the nodes from 1 to n in increasing order according to the following rule: choose as the next node one with a maximum number N(v), by letting n(v) denote the number of previously numbered neighbours of node v, if v is a continuous node, then N(v)=n(v), otherwise N(v)=n(v)+k, where k is the number of discrete nodes.

(d) compute the fill-in with respect to the order

Additional edges may be added to make the graph strongly decomposable.

(2) form cliques C

Cliques are maximal sets of nodes that are all connected to each other.

(3) order the cliques

By considering the highest numbered node in each clique, a clique order is obtained, C_1, C_2, ..., C_m, where the suffix i is an index to identify a particular clique such that C_1 has the lowest numbered node, C_m the highest numbered node. The intermediate cliques are numbered accordingly.

(4) determine residuals(R_i), separators(S_i) and parent cliques for each clique C_i

$R_i=C_i\cap(\cup C_j)$ j=1, ..., i-1. $R_1=\emptyset$ by definition.

$S_i = C_i - R_i$

If $S_i \subseteq C_j$ (j=1, ..., i-1), C_j is a parent clique of C_i.

(5) construct a clique tree T_C

(a) choose C_1 as the root.

It is a theorem [113] that the cliques of a decomposable mixed graph can be organised in a clique tree with at least one strong root.

(b) for C_i (i=2,..,m), a link is set up between C_i and its parent clique. If there are more than one parent cliques, always choose the one with the lower index number to be the parent clique.

The basic computational data structure is thus established. The representations of CG potentials on these cliques and separators are then calculated. It is straightforward to obtain the potential representation from the corresponding conditional probability tables. For discrete variables j with probability $p(j)$, the canonical characteristics representation will be $(g(j), 0, 0)$ where $g(j)=\log p(j)$. However, for a continuous variable C with conditional probability density specified by equation (4.1) in section 4.1, the corresponding CG potential defined as $(g(i), h(i), K(i))$ on the space $I \times C \times Y$ ($I \in \Delta$ and $C, Y \in \Gamma$) can be derived, where

$$g(i) = -\frac{\alpha(i)^2}{2\gamma(i)} - \frac{1}{2}log(2\pi\gamma(i)), \quad h(i) = \frac{\alpha(i)}{\gamma(i)}\begin{bmatrix} 1 \\ -\beta(i) \end{bmatrix}, \quad K(i) = \frac{1}{\gamma(i)}\begin{bmatrix} 1 & -\beta(i)^T \\ -\beta(i) & \beta(i)\beta(i)^T \end{bmatrix}.$$

The detailed proof can be found in Appendix C. Note that the dimension of a potential is based on the number of continuous variables involved in the potential. So $h(i)$ will be an N vector and $K(i)$ an N×N matrix where N denotes the number of continuous variables involved. The CG potentials are computed for all nodes and multiplied in tables for appropriate clique marginals with respect to the clique tree.

For computational purposes, all the potentials are represented in the form of canonical characteristics (g,h,K). Conversely, the corresponding moment characteristics {p,ξ,Σ} can be calculated by switch operations when we perform weak marginalisation operations. Initial probability distributions can be worked out on the clique tree. The marginal probability of each discrete node and the mean and variance of each continuous node can be calculated by performing marginalisations on the clique containing the node.

4.4.3. Enter Evidence

As we deal with mixed probabilistic models, there are two possible types of evidence: discrete and continuous. However, evidence observed is often not known with certainty. It is essential to handle uncertain evidence in probabilistic reasoning systems. We divide each type of evidence further into two groups: certain and uncertain evidence. The evidence will enter proper cliques which contain the observed node. The CG potentials on these cliques are transformed accordingly and are ready for propagation.

For certain discrete evidence, a particular state has to be specified. For example, suppose node D has n states, $D_1, ..., D_n$. If i th state of D is observed, so $P(D_i)=1$ and $P(D_j)=0$, j$\in$ {1,n} and j$\neq$i. The evidence is entered by multiplying $P(D)$ by the potentials on the cliques containing D.

For certain continuous evidence, a particular value is observed. However, things are a bit more complicated as evidence reduces the dimension of all vectors and matrices containing it. Thus all of these have to be reduced and the corresponding descriptions modified. Assume that continuous variable C takes a certain value y^*_C. The CG potential $(g(i), h(i), K(i))$ containing the continuous node C can be partitioned as:

$$h(i) = \begin{bmatrix} h_1(i) \\ h_C(i) \end{bmatrix}, \quad K(i) = \begin{bmatrix} K_{11}(i) & K_{1C}(i) \\ K_{C1}(i) & K_{CC}(i) \end{bmatrix}.$$

The new potential after absorbing the evidence y^*_C will be (g*(i), h*(i), K*(i)):

$$K^*(i)=K_{11}(i), \qquad h^*(i)=h_1(i) - y^*_C K_{C1}(i),$$

$$g^*(i)=g(i) + h_C(i)\, y^*_C - \frac{1}{2}K_{CC}(i)(y^{*2}_C).$$

We distinguish uncertain discrete evidence from uncertain continuous evidence in mixed models. Adopting a slightly different evidence absorption scheme, we will use the same message passing technique to propagate uncertain evidence. That is, to call "collect evidence" followed by a call to "distribute evidence" from the root clique. Here we explain the details of evidence absorption scheme for uncertain evidence.

For uncertain discrete evidence on D with n values the user will provided the estimation $(E_1, ..., E_n)$ by the user, where $E_i \geq 0$. First, the system will normalise them so that we have probabilities on D, i.e.

$(P_1, ..., P_n)$ where $P_i \geq 0$ and $\sum_{i=1}^{n} P_i = 1$. Then the CG potential representation of the uncertain evidence will be represented as $(\log P_i, 0, 0)$. If the marginal probability of D is P_D, the CG potential representation of the probability is $(\log P_{Di}, 0, 0)$. Finally, this uncertain evidence is absorbed in the clique containing node D. Assume that the clique has the CG potential $(g(i), h(i), K(i))$. The CG potential on that clique will then be updated as $(g^*(i), h^*(i), K^*(i))$. The calculations are:

$$g^*(i) = g(i) + \log P_i - \log P_{Di} ,$$

$$h^*(i) = h(i) , \qquad K^*(i) = K(i) .$$

Let C be a continuous node. Unlike certain continuous evidence which is a particular observation y^*_C of C, the uncertain continuous evidence on C can be specified as two parameters of the distribution of C: mean (m) and variance (v). Note that if the variance is zero, the evidence observed is certain. The uncertain evidence yields the canonical characteristics $(g^{new}, h^{new}, K^{new})$ where

$$g^{new} = -\frac{m^2}{2v} - \frac{1}{2} log(2\pi v) ,$$

$$h^{new} = \frac{m}{v} , \qquad K^{new} = \frac{1}{v} .$$

If the marginal CG potential representation of C, before observing the uncertain evidence, is $(g^{old}, h^{old}, K^{old})$, the potentials $(g(i), h(i), K(i))$ on the clique containing C will be updated to $(g^*(i), h^*(i), K^*(i))$ after absorbing the evidence, where

$$g^*(i) = g(i) + g^{new} - g^{old} ,$$

$$h^*(i) = h(i) + h^{new} - h^{old} ,$$

$$K^*(i) = K(i) + K^{new} - K^{old} .$$

4.4.4. Evidence Propagation

When evidence arrives concerning any particular node, its implication can be propagated to any other node through the clique tree without any global supervision to prevent inconsistencies. It involves only local computation between neighbouring cliques. Each clique can receive and pass on "messages" from its neighbours through separators between them [31]. The root clique issues a message "collect evidence" to its neighbours, who pass it to their neighbours until it reaches the leaf of the tree. Each clique then collects evidence from those neighbours further away from the root clique, using the fundamental operations:

marginalise, update and renew. Marginalisation is even more complicated as it is performed in two steps with intermediate transformations between the canonical and the moment characteristics. Moreover, the ambiguity between weak and "traditional" marginalisation has to be resolved.

If clique C_U absorbs from its neighbours C_W, and $\phi(S_{UW})$ is the strong marginal of $\phi(C_U)$, then C_U and C_W are consistent after absorption. The necessity to use clique trees with a strong root to obtain exact propagation results, is a consequence of this. All C_i and S_i satisfy the property (4.4) so that all potentials will be CG when propagating toward the strong root. Based on the local operation of absorption the propagation scheme can now be constructed exactly as for the discrete case.

When finally the root clique has collected evidence from its neighbours, it normalises its new potential when appropriate for evidence distribution. A message "distribute evidence" is issued. In the procedure of distribution, the same message passing operation is involved but working back through the tree. However, the potentials will not necessarily be CG. To understand this, note that the property (4.4) is not symmetric. This, in conjunction with the property that the CG-distribution is not closed under marginalisation, prompted Lauritzen to introduce the notation of weak marginalisation [34]. In essence, a mixture collapsing procedure, in which a mixture of Normals is replaced by a Normal with a mixture mean and mixture variance, is employed. In particular this gives us the correct updated probabilities of the states at any discrete node and the correct updated mean and variance of any continuous variables. The updated probability distribution for a discrete node or marginal mean and variance for a continuous node can be derived from these cliques. The propagation algorithm can be summaried as follows

```
evidencepropagation()
BEGIN
   absorb evidence
   choose a root clique R_CLIQUE of the clique tree
   messagepassing(R_CLIQUE)
END


messagepassing(R_CLIQUE)
BEGIN
   collectevidence(R_CLIQUE)
   normalise the CG potentials on R_CLIQUE when appropriate
   distributeevidence(R_CLIQUE)
END
```

```
collectevidence(CURRENT_CLIQUE)
BEGIN
    IF CURRENT_CLIQUE has children
    THEN FOR each child clique CH_CLIQUE of CURRENT_CLIQUE DO
            collectevidence(CH_CLIQUE)
    ELSE
        RETURN
    update CG potentials on separators of CURRENT_CLIQUE
    calculate the update ratio
    update CG potentials on CURRENT_CLIQUE
    RETURN
END
```

```
distributeevidence(CURRENT_CLIQUE)
BEGIN
    IF CURRENT_CLIQUE has children
    THEN FOR each child clique CH_CLIQUE of CURRENT_CLIQUE DO
            BEGIN
                update CG potentials on separators
                calculate the update ratio
                update CG potentials on CH_CLIQUE
                distributeevidence(CH_CLIQUE)
            END
    ELSE
        RETURN
END
```

Though only three basic operations are needed for computations in the clique tree, things are complicated due to the different kinds of tables and types of characteristics. Further, as tables are often different sizes, some control structure has to be provided for matching entries from different tables. Theoretically, tables are extended to be of the same structure, but in practice a separate description of the tables is held.

4.5. An Example

Most governments try to estimate crop yields for planning and other purposes. The total area sown with a crop is relatively easy to establish, but the average yield per hectare, which depends on meteorological, disease and pest factors, is more difficult to predict. One way of getting information on likely yields per hectare is to use the views of crop experts who are able to assess the factors involved and their possible effects on yield. Any such adjustment to the yield will affect the price in the market and the quantity available for export.

Suppose a crop expert believes that an important factor affecting autumn-sown cereal yields is the severity of winter. A very mild winter allows fungal diseases to be carried over from one season to another, leading to reductions in yield. A very cold winter kills diseases but may also lead to a loss of plants from frost damage. If plant loss is considerable then a crop may be re-sown with lower-yielding spring cereals.

For the purposes of this example we will assume that winter severity, the effects of disease and of frost, and the decision to re-sow, are binary discrete variables, while yield response, the price in the market and the quantity available for export are measured as continuous variables. The crop expert supplies the probabilities that the weather and other causal factors will occur. The yield response is measured in terms of expected percent change from that of the previous year. The price in the market and the quantity available for export are measured in terms of expected percent increase (positive value) or decrease (negative value).

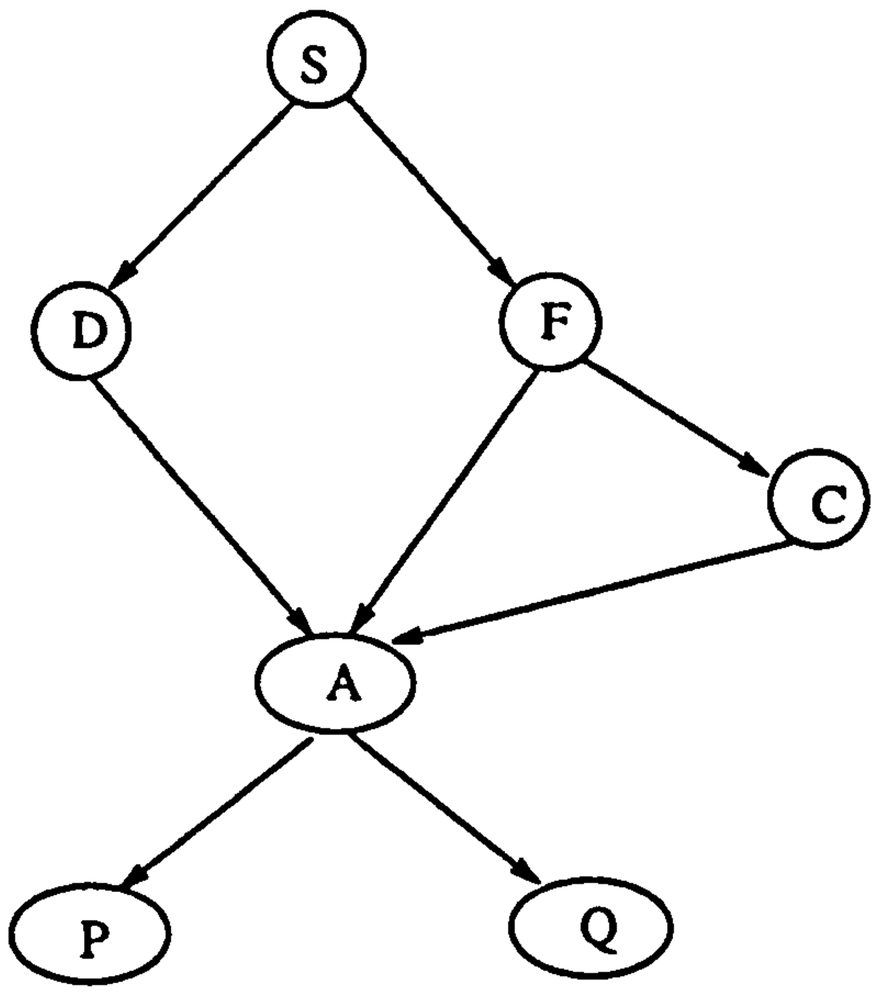


Figure 4.1. Graphical Structure of the Example

A structure which summarises the relations amongst the variables is shown in Figure 4.1, where circles represent discrete variables, also known as factors, ellipses denote continuous variables, and arrows indicate the dependencies between variables. For example, D represents the disease levels, while A represents the continuous variable, yield adjustment. The arrow between D and A indicates that the disease levels will affect the yield adjustment. The meaning of each node is given in Table 4.1.

Discrete Variables:

S: severity of winter,	s: severe,	¬s: mild;
D: disease levels,	d: high,	¬d: low;
F: frost damage,	f: high,	¬f: low;
C: spring crop sown,	c: yes,	¬c: no.

Continuous Variables:

A:	yield adjustment, distributed normally,
P:	market price, distributed normally,
Q:	quantity available for export, distributed normally.

Table 4.1. Meaning of Variables in the Example

The associated prior and conditional distributions are listed in Table 4.2 along with some specimen parameter values. For example, the probability that the winter has had a severe effect on the crops, p(S = severe) or p(s) is 0.3. The probability that disease levels will be high, p(D = high | S = severe), p(d | s) is 0.2. Similarly, the other conditional probability estimates and their interpretations are listed at the top of Table 4.2.

p(s)=0.3	(=> p(¬s)=0.7)
p(d | s)=0.2	disease levels low if severe
p(d | ¬s)=0.6	winter, high if mild;
p(f | s)=0.7	frost damage high if severe
p(f | ¬s)=0.2	winter, low if mild;
p(c | f)=0.8	spring crop likely to be sown if
p(c | ¬f)=0.1	frost damage, unlikely if not.

F(A | d, f, c) = N(-1,1)
F(A | d, f, ¬c) = N(-3,1)
F(A | d, ¬f, c) = N(1,1)
F(A | d, ¬f, ¬c) = N(-1,1)

F(A | ¬d, f, c) = N(1,1)
F(A | ¬d, f, ¬c) = N(-1,1)
F(A | ¬d, ¬f, c) = N(3,1)
F(A | ¬d, ¬f, ¬c) = N(1,1)

(Note: the means for the yield adjustments have been determined according to the following rules and combined by addition; the variances are all set to 1

d => A = -1 ¬d => A = 1;
f => A = -1 ¬f => A = 1;
c => A = 1 ¬c => A = -1)

F(P | a) = N(-2a, 1) F(Q | a) = N(3a, 1)

Table 4.2. Numerical Assessments of the Example

Also in Table 4.2 are suggested parameter values for the mean and variance of the continuous yield response variable A. Separate values are given for each combination of values of those discrete variables which directly affect yield. In each case the distribution is assumed to follow the Gaussian distribution, that is, estimates are expected to be symmetrically distributed about the true value with a specified variance, i.e. N(expected mean, variance). Note, there are no continuous parents of A. The crop expert supplies estimates of the expected mean response and variance for each combination of disease levels, frost damage and crop re-sow possibilities. For example, if disease levels are high (D=d), frost damage is high (F=f) and the crop has not been re-sown (C=¬c) then the expected yield change is -3% with an estimated standard deviation of 1%.

The price in the home market and the quantity available for export depend on the yield adjustment. It is suggested by an expert that an adjustment in yield of amount a% will give rise to a price adjustment in *£/unit weight* which is normally distributed with mean -2a and variance 1 and to an adjustment in the quantity by weight available for export which is normally distributed with mean 3a and variance 1. Thus, for example, an adjustment in yield of 2% (a=2) will give rise to a price adjustment, normally distributed, mean -£4/*unit weight*, variance 1, and to an adjustment in the quantity by weight available for export, normally distributed with mean 6%, variance 1.

The dependency structure may then be decomposed using the algorithm described in section 4.4.2. Additional undirected links are added to make the original structure triangulated as displayed in Figure 4.2, where solid lines represent original edges and dashed lines represent the marriage of parents. No fill-in is needed in this case. The numbers in the graph show the result of applying the search algorithm described in section 4.4.2. Cliques and separators are derived from the strongly decomposable graph, see Table 4.3. There are four cliques in our example, {S,D,F}, {D,F,C,A}, {P,A} and {Q,A}. A clique tree, shown in Figure 4.3, where ellipses represent cliques and squares represent separators, is built from Table 4.3. The clique {S,D,F} is chosen as the root in the clique tree.

Marginal probabilities for the factors and means and variances for the continuous variables may be evaluated using the rules of conditional probability and are given in Table 4.4 below.

When evidence is observed, message passing is used to update belief. The evidence collection and distribution procedures for our example are displayed in Figure 4.4 and Figure 4.5 respectively. The

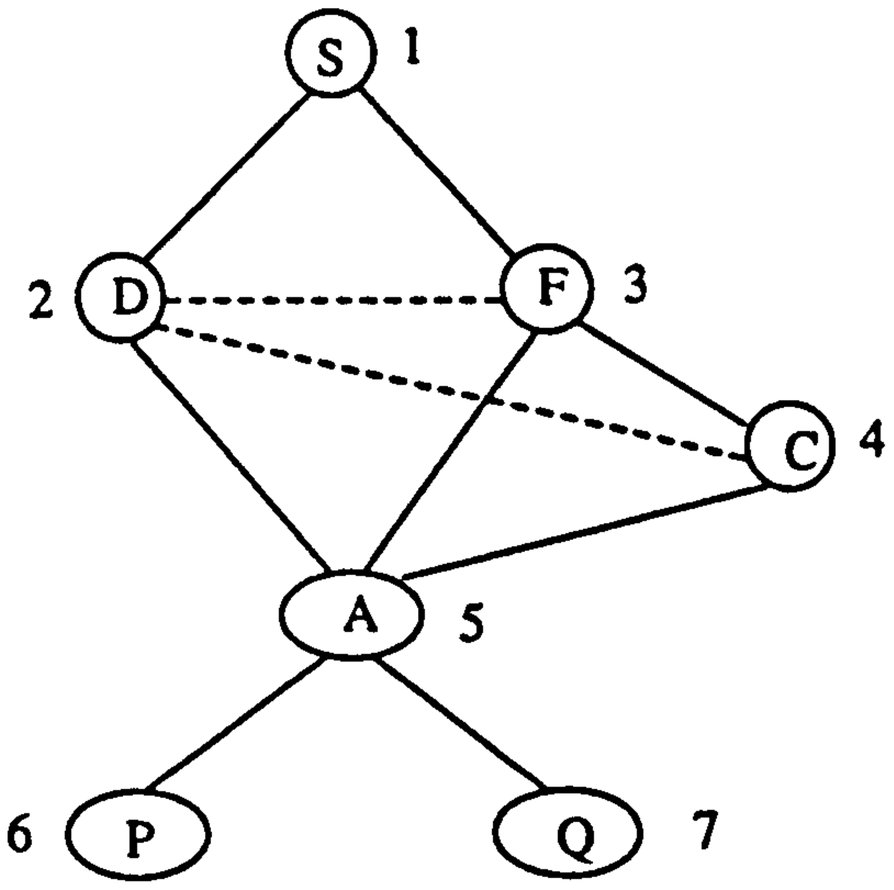


Figure 4.2. Decomposed Graph of Our Example

Clique	Members	Residuals	Separators	Possible Parent Cliques
1	S,D,F	S,D,F	∅	
2	D,F,C,A	C,A	D,F	1
3	A,P	P	A	2
4	A,Q	Q	A	2,3

Table 4.3. Cliques and Separators of Our Example

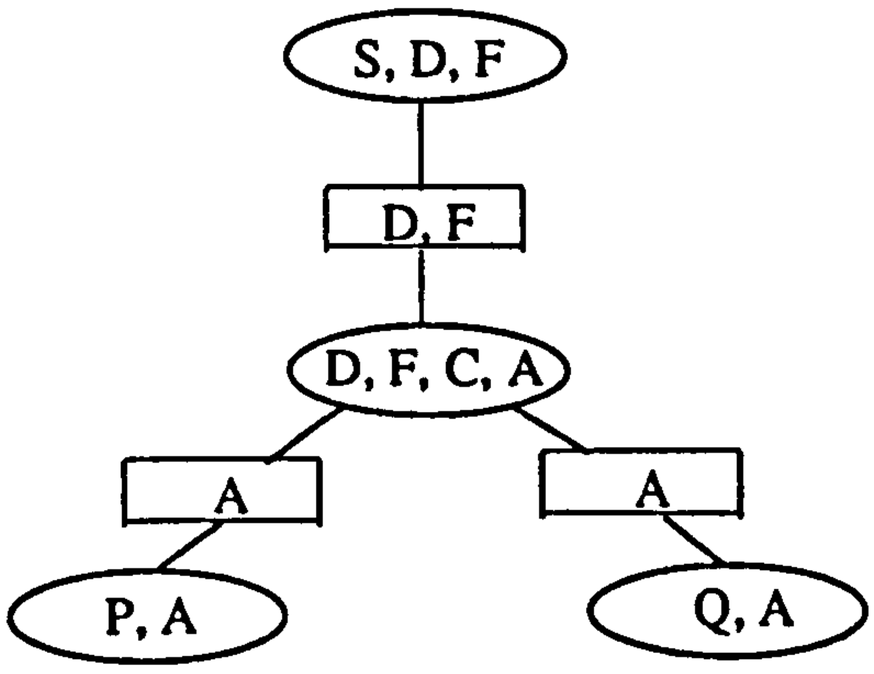


Figure 4.3. The Clique Tree of Our Example

p(s)=0.3		p(d)=0.48
p(f)=0.35		p(c)=0.345
Variable	Mean	Variance
A	0.03	2.4375
P	-0.06	10.7500
Q	0.09	22.9375

Table 4.4. Initial Marginal Probabilities, Means and Variances

dashed arrows indicate the direction of message passing and the numbers above those arrows indicate the order of operations. For example, all level 1 message passing must be carried out before level 2 takes

place.

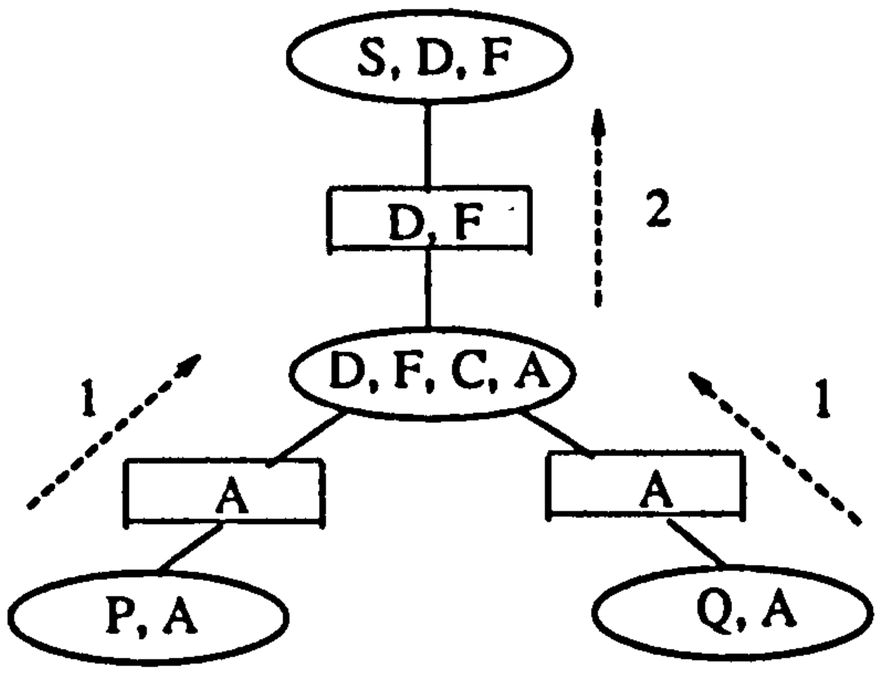


Figure 4.4. Collect Evidence Procedure

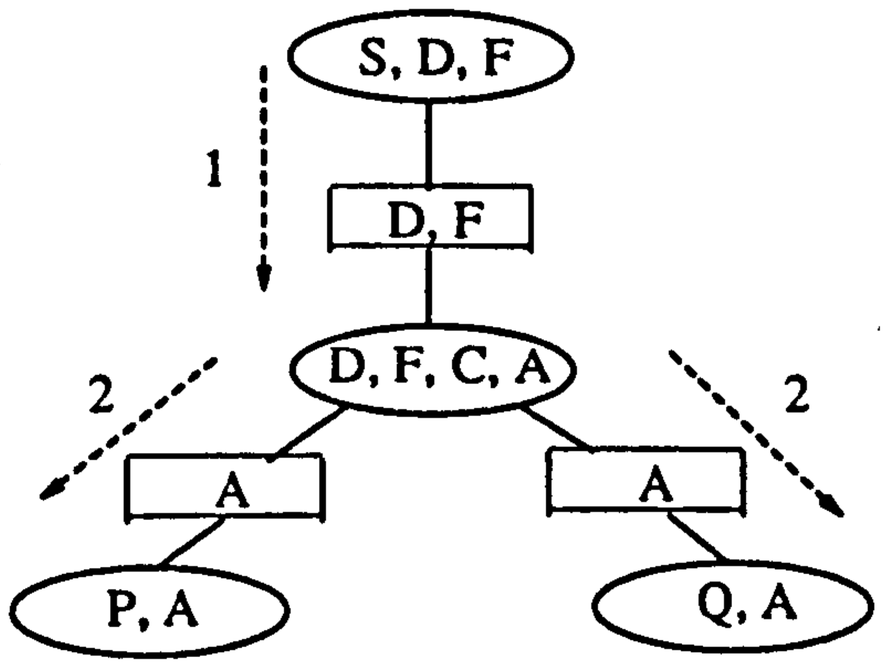


Figure 4.5. Distribute Evidence Procedure

It is possible, now, to use the propagation algorithm to answer, for example, the following question 'If the winter is severe, by how much should the predicted yield be adjusted?' The updated probability distributions, showing the revised probabilities for disease levels, frost damage and the sowing of spring crop and the revised means and variances for the yield adjustment, market price and quantity available for export, are shown in Table 4.5. Note, for example, that yield would be adjusted by 0.38%; price by -£0.76 per unit weight and the quantity available for export by 1.14%.

Further queries which might be answered include: 'The quantity available for export has risen by 2%, what adjustment might be made to the price on the home market?'. The updated belief is displayed in Table 4.6.

p(s)=1 p(f)=0.7		p(d)=0.2 p(c)=0.59
Variable	Mean	Variance
A	0.38	2.2716
P	-0.76	10.0864
Q	1.14	21.4444

Table 4.5. After Winter Was Severe

p(s)=1 p(f)=0.6819		p(d)=0.0742 p(c)=0.6480
Variable	Mean	Variance
A	0.6784	0.1041
P	-1.3568	1.4164
Q	2	-

Table 4.6. After Winter Was Severe and the Export Quantity Has Risen by 2%

Note, here, that yield would be adjusted by 0.68%, price by -£1.36 per unit weight.

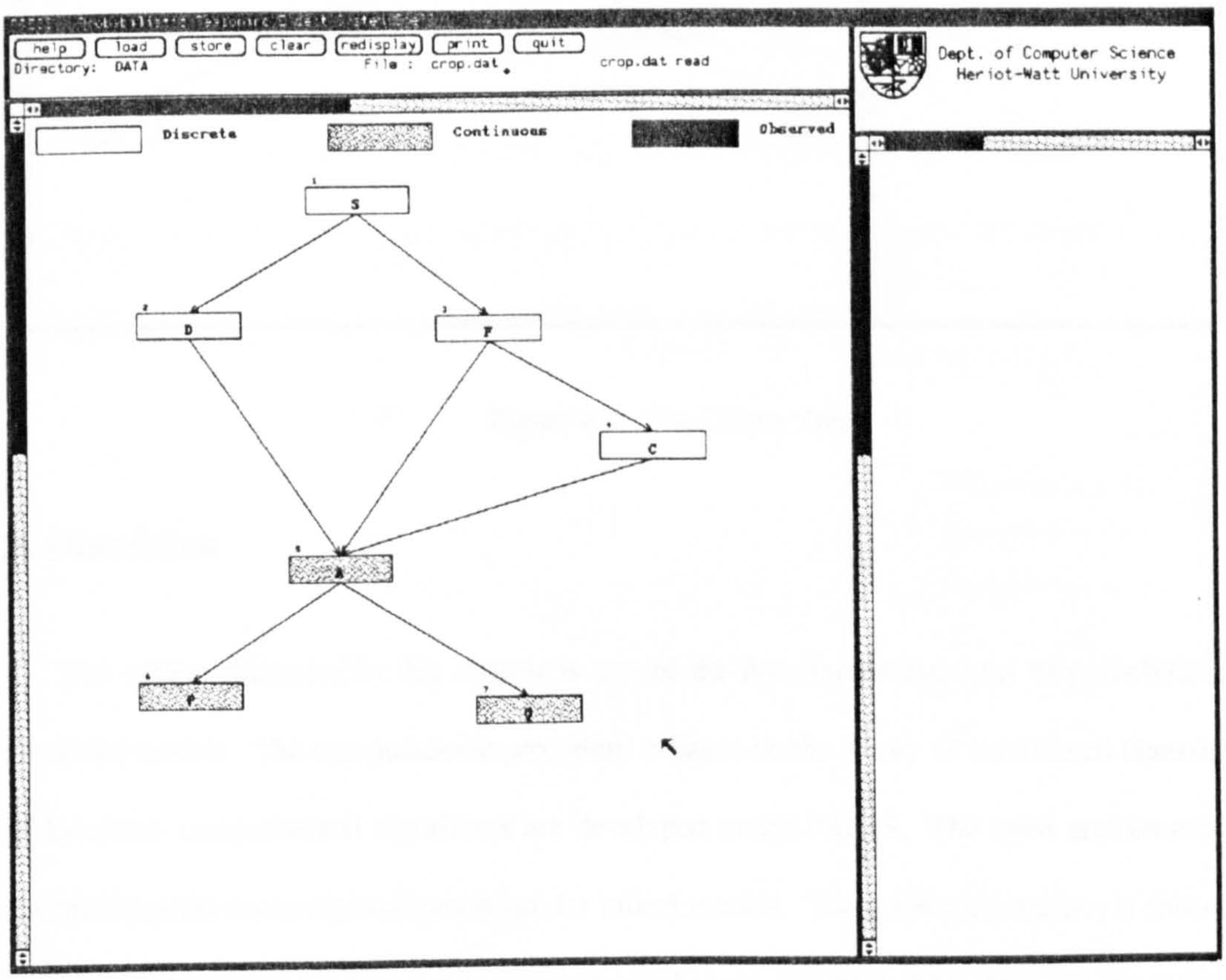


Figure 4.6. Graphical Structure of the Agricultural Example

Screendumps from using the extended PRESS to deal with the agricultural example are given as follows. Figure 4.6 shows the Bayesian belief network for the example. A clique tree of the example is presented in Figure 4.7. The initial marginal probabilities of discrete variables and means and variances of

continuous variables are displayed in Figure 4.8. The updated probabilities and means and variances after winter was severe and the export quantity has risen by 2% are given in Figure 4.9 and Figure 4.10, respectively.

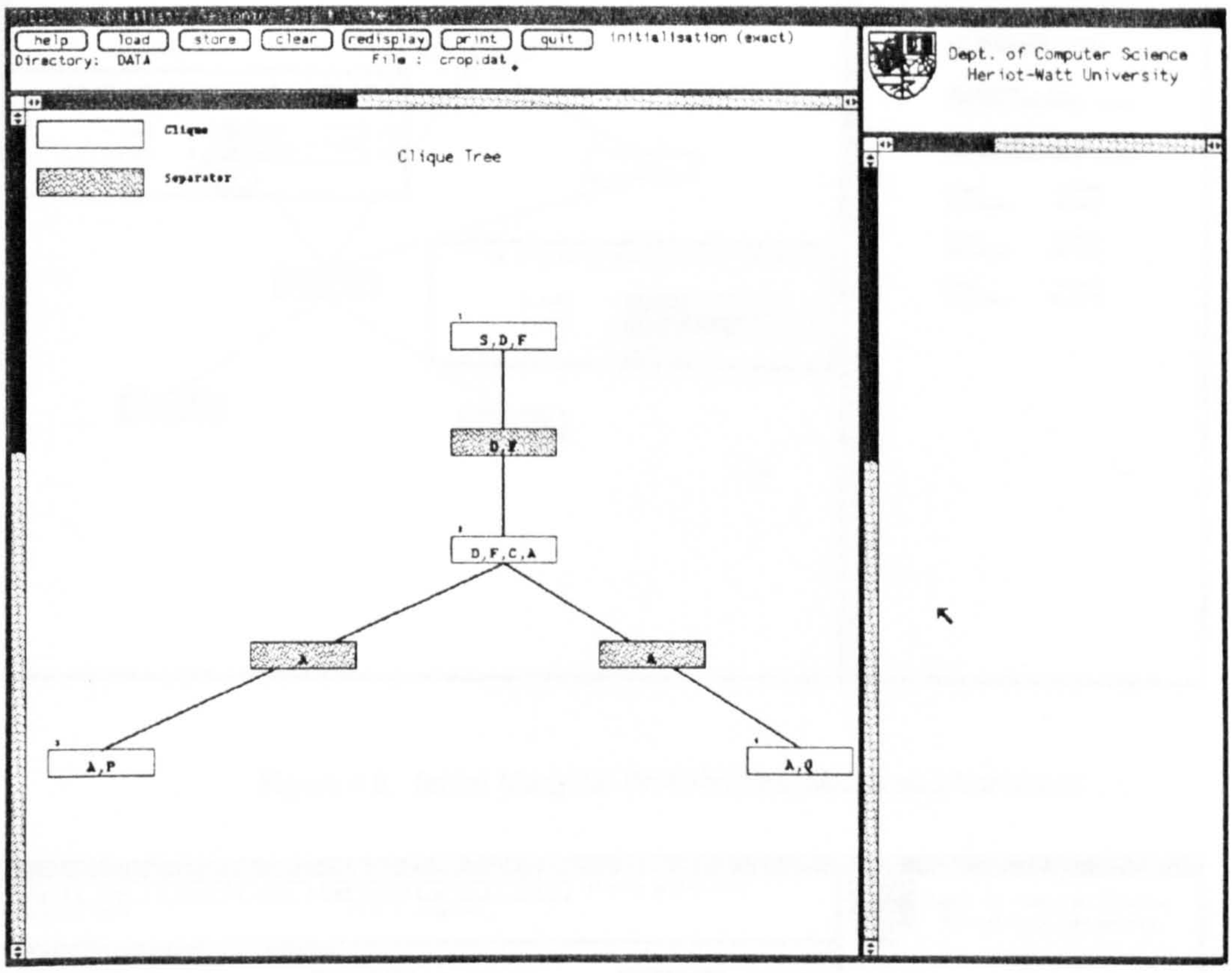


Figure 4.7. The Clique Tree

4.6. Discussion

The system discussed in this chapter is one of the first implementations of probabilistic reasoning with mixed models. The computational procedure is based on the theory of conditional Gaussian networks [34] but these computational algorithms are developed under PRESS. The open architecture of PRESS accommodates the computational procedure for mixed models. The graphical interface is enhanced to cope with the specification of a mixture of continuous and discrete variables. The agricultural example discussed above illustrates the extended PRESS system and demonstrates how an expert's opinions are incorporated into a statistical procedure for assessing evidence.

The computational procedure is based on the following assumptions [34,109]. First, no continuous variables have discrete children. Second, the conditional distribution of a continuous variable given other

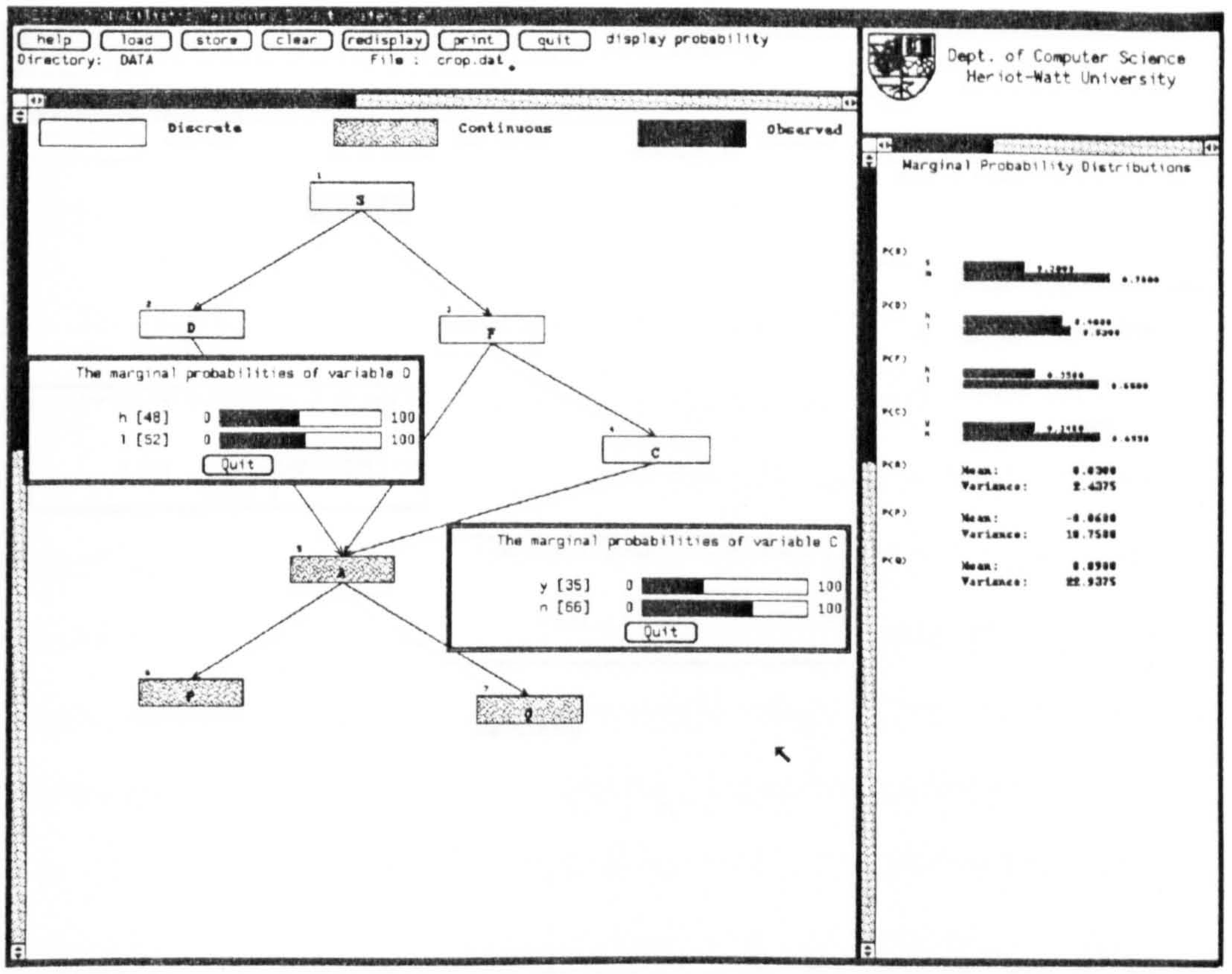


Figure 4.8. Initial Marginal Probabilities, Means and Variances

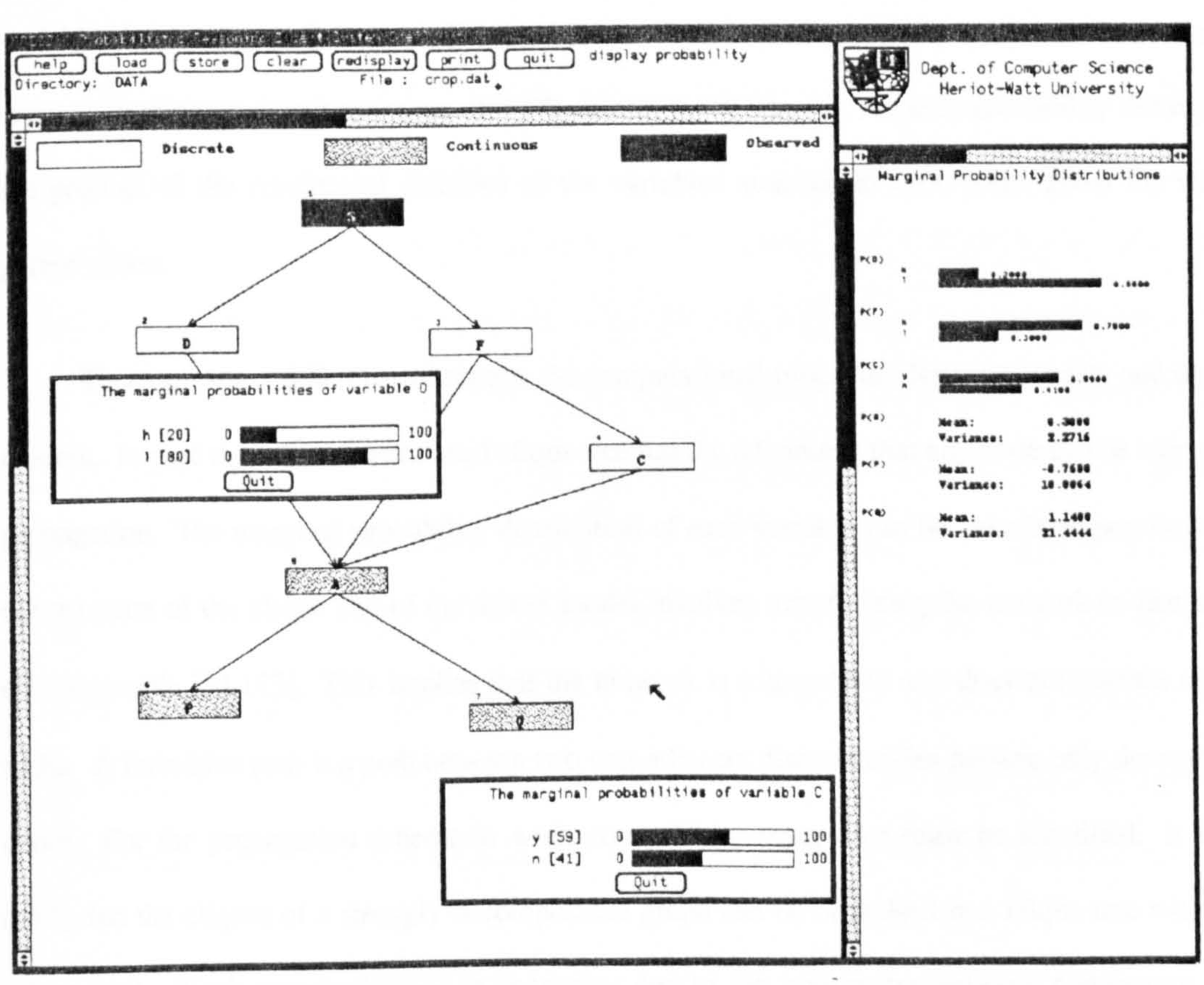


Figure 4.9. After Winter Was Severe

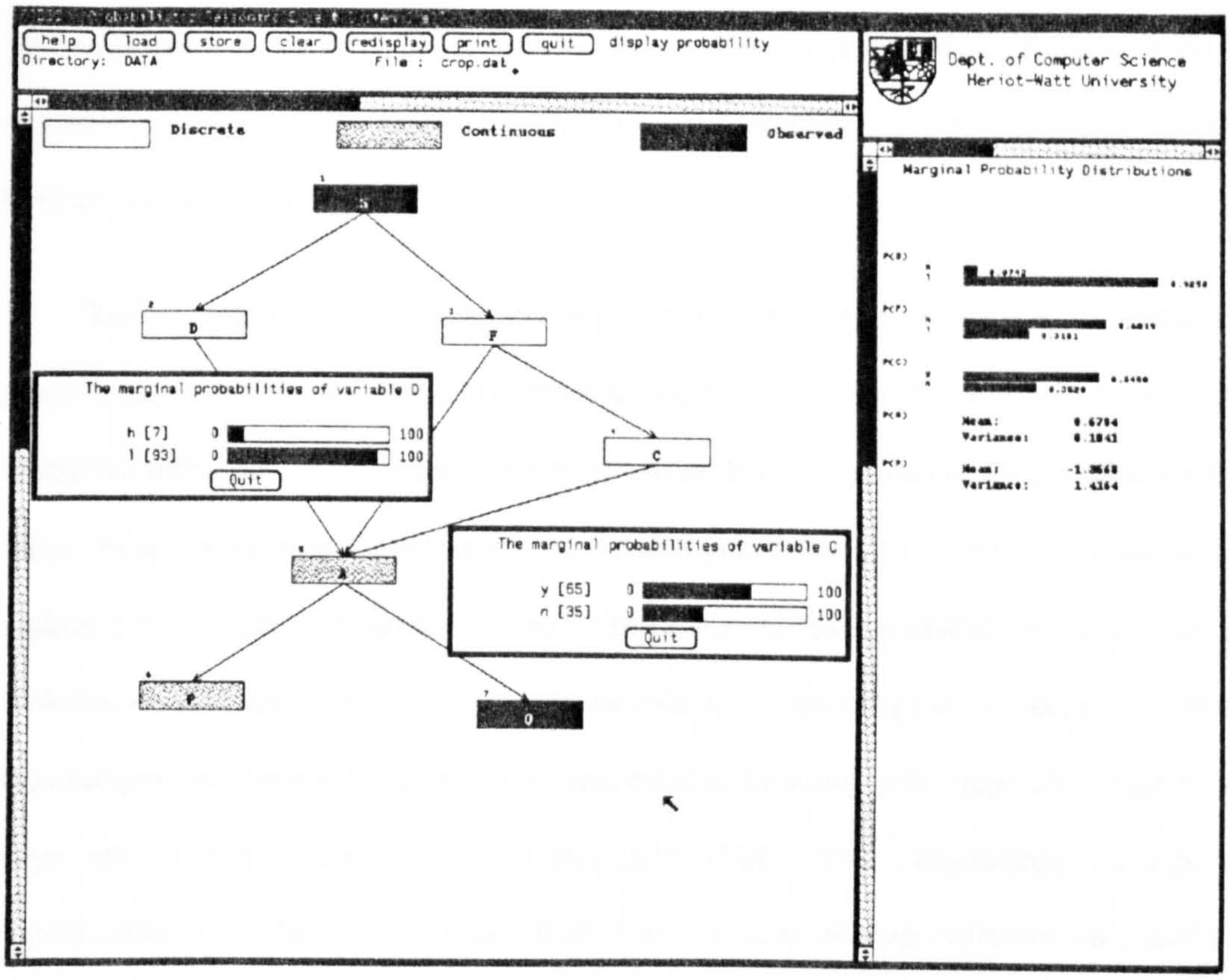


Figure 4.10. After Winter Was Severe and the Export Quantity Has Risen by 2%

variables is conditional Gaussian. Third, all interactions between variables are linear functions. That will ensure the resultant distributions are still CG-distributions. Fourth, the joint probability density is equal to the product of the conditional densities of the variables attached to each node, given the states at their parent nodes.

There are some differences between the computational procedure for pure models and that for mixed models. In pure models, the associated clique tree has the advantage that any node can be used as a root for propagation. The marginal probability distribution of each variable can be calculated precisely. However, construction of the clique tree of the mixed model involves transforming the network so that it is strongly decomposable [34,113]. This implies that the network is triangulated and does not contain any forbidden paths. A forbidden path is a path between two non-adjacent discrete nodes passing only through continuous nodes. For the propagation scheme to work correctly, a *strong root* must be identified. It is a theorem [113] that the cliques of a strongly decomposable graph can be organised in a clique tree with at least one strong root. *Weak marginalisation* is necessary due to the asymmetry between continuous and discrete variables and CG-distribution is not closed under marginalisation. So the approximation of the marginal

distribution of a clique by a CG-distribution whose moment characteristics agree with the true marginal moments is carried out. For example, the true marginal distribution of yield adjustment will be a mixture of 8 normal distributions.

Various approximate solutions to the problems described in section 4.4 are discussed [34]. If the graph is such that a discrete factor is dependent on a continuous variable, one possibility is to use the ideas of logistic regression. It is then possible to approximate the logarithm of the probabilities for the discrete factors by a second order Taylor expansion. Another possibility is to discretise the continuous variable and replace it by a categorical variable whose various categories have probabilities equal to the corresponding probabilities calculated from the continuous variable. When applying the evidence propagation algorithm to the multiprocess dynamic linear model, it turns out that the strongly decomposable graph for the model has large cliques and is computationally infeasible [114]. The computational savings of using CG representations may be lost. An approximation involving weak marginalisation on a modified CG clique tree may appear to be adequate [34,114]. In these cases, and other approximations discussed by Lauritzen [34], the quality of the approximations is an open question.

Unfortunately, the computation of the marginal density functions of continuous variables is generally forbiddingly complex in mixed models. Stochastic simulation is being investigated as a way of obtaining approximate estimates of marginal densities [115]. Recent investigation shows that the stochastic simulation algorithm[32] can be extended to models with both discrete and continuous variables [115]. Applying stochastic simulation enables us not only to estimate the marginal probabilities, means and variances of variables, but also to obtain an idea of the *shape* of the distribution for continuous variables. With stochastic simulation however, an idea of the marginal densities is obtained from the simulated values. The computations themselves are straightforward, and consideration of each node in a network involves only its neighbours and its childrens' parents. The simulated values can be plotted for a rough idea of the shapes of the density functions. Alternatively, kernel density estimation techniques are applied to the simulated values of the continuous variable to obtain estimates of the marginal probability densities.

Another interesting problem is how to combine Bayesian belief networks and time series analysis to model dynamical world. Probabilistic reasoning system for mixed models could provide an opportunity for investigation and further research.

4.7. Concluding Remarks

In this chapter, a computational procedure for dealing with mixed probabilistic models is designed and developed. This enables a natural representation of various entities measured as real numbers. The model is based on directed Markov fields of CG-distributions [76,116]. Given the structure which reflects the dependence among variables and the corresponding conditional probability tables, the marginal probabilities of discrete variables and the means and variances of continuous variables are calculated and updated given evidence based on "local computation". However, the essential difference between the model described here and that of the previous chapter is because of the asymmetry between the continuous and discrete variables and because CG-distributions are not closed under marginalisation in general. For the model specification to be exactly faithful, the graphical knowledge representation behind the model has to satisfy the constraint that no continuous nodes have discrete children. Similarly the propagation algorithms give exact results only if the undirected graph is strongly triangulated. The true marginal distribution at any cliques typically would be a mixture of CG-distributions and not a CG-distribution itself. So all we can get is the correct updated marginal probabilities and the correct updated mean and variance of any continuous variable.

An extension of PRESS including this computational module has been made. The local computations are fulfilled in the system if we make a strongly decomposable graph. As a result, it leads to a natural extension of PRESS. Moreover, the extension allows a mixture of discrete and continuous variables to be modelled and manipulated and enhances the applicability of the system.

It should be mentioned that some additional approximations are possible [34]. One concerns the case where continuous nodes in Bayesian belief networks are allowed to have discrete child nodes. Another concerns relaxation of the requirement of the triangulation to be strong. Both possibilities may introduce some errors but these will probably be of minor importance compared to the uncertainty involved in the model construction process.

Chapter 5

LEARNING IN BAYESIAN BELIEF NETWORKS

The adequacy of a model's predictions depends both on its qualitative structure and the assessed parameters. However, no matter how a model has been constructed initially, it is important to have methods for the possibility of updating and improving the model through experience. How to obtain the necessary probabilities is frequently a major concern in applying Bayesian belief networks. The assumption that assessed parameters, conditional probability tables on appropriate set of variables, are precisely defined by domain experts or from past data, is made in the development of probabilistic reasoning systems. This assumption is not realistic. The probabilities derived from subjective assessments or specific data are subject to inevitable imprecision [117]. An extensive literature on human judgement has identified cognitive biases and mental heuristics that tend to distort human judgements about uncertain events [118]. Any expert system should learn by experience in order to overcome the inevitable limits on the knowledge of those who initially developed it.

Bayesian belief networks are usually constructed by knowledge engineers working with experts in the domain of interest. This procedure is called "knowledge acquisition". However, it is well known that knowledge acquisition is a bottleneck in developing expert systems. Traditional approaches to knowledge acquisition have included psychological techniques for interviewing experts and automatic production of classification-oriented expert systems from examples. Such approaches have had a limited range of successful applications. Furthermore, experts may have only partial knowledge about the domain. On the other hand, computer-based information processing has changed dramatically. Data is being generated rapidly, such as that obtained from medical surveys. Therefore, methods of exploring databases and constructing Bayesian belief networks automatically from data, thus bypassing knowledge acquisition, are important.

Under the Bayesian approach, the learning in Bayesian belief networks separates into two highly related tasks, *parameter learning*, that is to revise the numerical parameters (i.e. conditional probabilities) for a given network topology as new cases arrive, and *structure learning*, that is to identify the topology of the network from a database.

The aim of this chapter is to extend PRESS to accommodate different learning techniques and carry out experiments on these learning techniques. A simulated database and a real database of abdominal pain patient records [38] will be used for the experiments. PRESS provides a computational framework for expressing imprecision and revision of conditional probabilities, and inference as data accumulates.

First, a parameter learning method proposed by Spiegelhalter and Lauritzen, namely sequential learning, is described [36]. A simplest case, when conditional probability distributions can be expressed as Dirichlet distributions is investigated. The method has been applied to the medical database. A comparison between the system with the learning function and that without the learning function is made.

Next, we attempt to address the structure learning issue. We would like to form a model in a semi-automated process by extracting the underlying topology directly out of data. Our focus will be on polytree models. The polytree algorithm proposed by J. Pearl is discussed and implemented [29]. Experiments with the simulated database and the medical database are conducted. As a result a Bayesian belief network for diagnosing abdominal pain is constructed and used for predicating diagnoses based on the system PRESS when new cases arrive. The performance of the model extracted has been compared with other classification methods.

Finally, the problems associated with learning in Bayesian belief networks and the experiments will be discussed.

5.1. Parameter Learning

It would be possible to avoid the assumption that the assessed conditional probabilities are precisely defined and to express our doubt on these probabilities by adopting a Bayesian learning approach [36,26,119]. The basic idea is to represent the imprecision of conditional probabilities explicitly as parameters. These parameters with uncertainty attached to them are defined in the form of probability

distributions specified by domain experts or past data. The new case is then to sharpen the posterior distributions of parameters and cause them to be nearer the true values of the conditional probabilities. The mechanism for progressively estimating improved parameters plays a very important role in parameter learning.

We emphasise the computation of a complete joint probability distribution conditional on observed evidence $P(V|E)$ in probabilistic reasoning systems. It is not difficult to calculate the posterior probability $P(V|E)$ if we can specify the distribution $P(V)$, for example, as the product of conditional probabilities specified on Bayesian belief networks [25,28,33]. However, the domain experts or past data can only specify the joint probability distribution $P(V|\theta)$ over a parameter space θ. The parameter θ is assumed to be contained in a known priori distribution $P(\theta)$. This prior distribution captures our knowledge or experience about what is likely and what is not, and is relevant to the processing of future cases. In short, the parameter θ plays the role of a "dynamic knowledge base". The rest of our knowledge about θ is supposed to be a set of new cases E. Therefore, the basic task of parameter learning is to calculate the posterior distribution $P(\theta|E)$. The additional knowledge of E to the parameter θ will be stored for processing the next case.

For computational purposes, we assume that new cases are drawn independently and are exchangeable over a sequence so that we can concentrate on operations in a single case and work with each case separately in the light of a series of cases. In other words, the model is constant over that period. Consider processing a new case E_j, $E_j \in E$. Three basic operations are involved: dissemination of experience, propagation of evidence and retrieval of new information [36]. The procedure can be repeated in the same manner as new cases arrive. This process is known as "sequential learning" [36].

In a Bayesian belief network setting, it is reasonable to partition the space θ into a set of small spaces θ_i concerning each node V_i and assume θ_i is independent to each other, that is,

$$P(\theta) = \prod_{i=1}^{n} P(\theta_i),$$

where n is the number of nodes in the graph. The parameter θ_i completely specifies the relationship between a node V_i and its parent nodes $PA(V_i)$. In fact, each conditional probability table attached to node V_i is determined uniquely by the parameter θ_i. In other words, we consider the conditional probability

$P(V_i \mid PA(V_i))$ itself as a random variable over the parameter space θ_i and θ_i is in the form of a probability distribution $P(\theta_i)$. To emphasise the dependency of $P(V_i \mid PA(V_i))$ upon θ_i explicitly, we will write the conditional probability of V_i given its parent nodes as $P(V_i \mid PA(V_i), \theta_i)$ and the specification of $P(V_i \mid PA(V_i), \theta_i)$ is determined by $P(\theta_i)$. Therefore, $P(V_i \mid PA(V_i), \theta_i)$ and $P(\theta_i)$ are needed to be specified for each variable V_i in the model.

The joint probability distribution $P(V \mid \theta)$ can be written as follows due to the conditional independence reflected in the model:

$$P(V \mid \theta) = \prod_{i=1}^{n} P(V_i \mid PA(V_i), \theta_i) .$$

The joint probability distribution on V and θ is then calculated as

$$P(V, \theta) = \prod_{i=1}^{n} P(V_i \mid PA(V_i), \theta_i) P(\theta_i) .$$

From the above expression it is clear that the parameter θ_i may be considered as another parent node of V_i in the Bayesian belief network. These θ_i parameters form a level above the Bayesian belief network. This level represents our experience and summary of past cases [36].

Given the structure, and $P(V_i \mid PA(V_i), \theta_i)$ and $P(\theta_i)$ specified for each node V_i, our task now is to calculate the posterior distribution $P(\theta \mid E_j)$ when a new case E_j is obtained. To make the computations simpler, different assumptions are made for different operations. Independence of each parameter θ_i over node V_i allows the dissemination operation to be carried out locally, that is, for each variable V_i, we apply

$$P(V_i \mid PA(V_i)) = \int P(V_i \mid PA(V_i), \theta_i) P(\theta_i) d\theta_i .$$

We get the expectations of the conditional probabilities $P(V_i \mid PA(V_i), \theta_i)$ for the currently assumed value attached to each node V_i. The L-S algorithm developed before can be used directly at the stage of evidence propagation [25,28]. In the retrieval operation, the following calculation is performed:

$$P(\theta_i \mid E_j) = \sum_{V_i, PA(V_i)} P(\theta_i \mid V_i, PA(V_i), E_j) P(V_i, PA(V_i) \mid E_j) .$$

Since θ_i is conditionally independent of E_j given V_i and $PA(V_i)$, thus

$$P(\theta_i \mid E_j) = \sum_{V_i,\, PA(V_i)} P(\theta_i \mid V_i, PA(V_i)) P(V_i, PA(V_i) \mid E_j) .$$

It is clear that there is a mixture distribution for the parameter θ_i if V_i and $PA(V_i)$ are not observed. To simplify the retrieval operation, the individual parameter θ_i for node V_i is further partitioned into a set of $\theta_i{}^+$ which is conditional on each possible configuration of its parent set $PA(V_i)^+$. These parameters $\theta_i{}^+$ for node V_i are assumed independent given any particular combinations of its parents. One consequence of this assumption is that each conditional probability distribution under a configuration of the parent nodes can be individually updated in the light of new cases E_j. The updating procedure for the parameter $\theta_i{}^+$ is carried out using the following operation:

$$P(\theta_i{}^+ \mid E_j) = \sum_{V_i} P(\theta_i{}^+ \mid V_i, PA(V_i)^+) P(V_i, PA(V_i)^+ \mid E_j) + P(\theta_i{}^+)\,(1 - P(PA(V_i)^+ \mid E_j)) .$$

As a result, the retrieval operation is achieved locally.

Having seen the possible operations on $P(\theta_i)$, the distribution on θ_i must be chosen carefully based on the following considerations:

1. it should be easily obtained from domain experts or past data;

2. it should be representationally simple;

3. the operations on $P(\theta_i)$ should be computationally manageable.

Spiegelhalter and Lauritzen [36] give a range of possibilities to specify such probabilistic models. In the next section we will consider the situation where the parameter θ_i follows a Dirichlet distribution [36] and $P(\theta_i)$ can be estimated from a database. The operations of dissemination and retrieval on a Dirichlet distribution will be discussed.

5.1.1. Modelling with Dirichlet Distributions

When we estimate probability values from a database, we often make the assumption that the relative frequencies represent the corresponding probabilities. However, unknown probabilities can be considered as random quantities and specified by a distribution as we discussed in the preceding section. It is usual to use a Dirichlet prior distribution as a conjugate form in this case (Note Beta distribution $B(a,b)$ is a special

case of Dirichlet distribution for a binary random variable, with mean $\frac{a}{(a+b)}$ and variance $\frac{ab}{(a+b)^2(a+b+1)}$).

Suppose that V_i is a discrete variable with m possible states (V_{i1}, ..., V_{im}). Consider a particular conditional probability distribution of V_i conditional on a configuration of $PA(V_i)^+$, that is, $P(V_i \mid PA(V_i)^+, \theta_i)$. For simplicity, we shall denote this distribution as τ_i. Therefore, τ_i has m possible values $\{\tau_{i1}, ..., \tau_{im}\}$, corresponding to all the possible states of V_i. Assume that the parameter θ_i has a Dirichlet distribution ε^+ with m parameters $\{\varepsilon^+_1, ..., \varepsilon^+_m\}$. So it is equivalent to having seen τ_{i1} occur ε^+_1 times, τ_{i2} occur ε^+_2 times, ..., and τ_{im} occur ε^+_m times in $\sum_{j=1}^{m}\varepsilon^+_j$ total occurrences. The size of $\sum_{j=1}^{m}\varepsilon^+_j$ is very important because it represents the uncertainty in the probability value. Basically, the larger the value of $\sum_{j=1}^{m}\varepsilon^+_j$, the more confident we are in the these probabilities. So we have a m-dimensional Dirichlet distribution of the form:

$$P(\tau_i \mid \varepsilon^+) \propto \prod_{j=1}^{m}(\tau_{ij})^{\varepsilon^+_j-1}, \qquad \text{for } \tau_{ij}>0,$$

where $\prod_{j=1}^{m}(\tau_{ij})^{\varepsilon^+_j-1}$ can be denoted as $D[\varepsilon^+_1, ..., \varepsilon^+_m]$. Note that the number of parameters necessary to specify a distribution over the possible conditional probabilities is m. The distribution specified may be interpreted as representing past experience as a contingency table of counts of past cases. The updating procedure consists of modifying the counts as new cases are being observed.

Let us consider the basic operations on this particular distribution. The dissemination operation is very simple in this case. The conditional probability $P(V_{ik} \mid PA(V_i)^+)$ (k$\in$[1,m]) is calculated from:

$$P(V_{ik} \mid PA(V_i)^+) = \int P(V_{ik} \mid PA(V_i)^+, \theta_i) P(\theta_i) d\theta_i = \frac{\varepsilon^+_k}{\sum_{j=1}^{m}\varepsilon^+_j}.$$

This is the conditional probability used for initialisation and evidence propagation. On the other hand, the retrieval operation can be very complex. In general, having observed E, we shall have

$$P(\tau_i \mid E) = \sum_{j=1}^{m} D[\varepsilon_1^+, ..., \varepsilon_j^+ + 1, ..., \varepsilon_m^+] P(V_i, PA(V_i)^+ \mid E)$$

$$+ D[\varepsilon_1^+, ..., \varepsilon_m^+](1 - P(PA(V_i)^+ \mid E)) . \qquad (5.1)$$

If we have observed both V_i and $PA(V_i)^+$ in the new case E, the retrieval is straightforward. 1 is added to the relevant parameter, corresponding to an additional example in our memory. For example, if V_i is observed to be in the state V_{ij}, we have

$$\tau_i \mid E \sim D[\varepsilon^+_1, ..., \varepsilon^+_j + 1, ..., \varepsilon^+_m] .$$

However, if V_i has not been observed, the retrieval operation involves dealing with a mixture of m Dirichlet distributions. A number of methods exist for approximating such a mixture by a single Dirichlet distribution after each observation, providing a conjugate prior for the next case [120,121]. Thus, in the general formulation, we assume that the expression (5.1) may be approximated by a Dirichlet distribution with the revised parameters $\{(\varepsilon^+_1)^*, (\varepsilon^+_2)^*, ..., (\varepsilon^+_m)^*\}$, where

$$(\varepsilon^+_j)^* = \varepsilon^+_j + P(V_{ij}, PA(V_i)^+ \mid E), \qquad j = 1, ..., m .$$

However, there are some problems with such approximations, see further discussion of this issue in Spiegelhalter and Lauritzen [36].

5.1.2. A Medical Database

There is a database for 6,387 patients who were admitted to hospital suffering from acute abdominal pain†. Each of these patients had their symptoms noted together with their diagnosis given by the doctors. Each record describes 33 symptoms and a diagnostic group. In total there are nine most common abdominal pain diseases. They are Appendicitis (APP), Diverticulitis (DIV), Perforated Peptic Ulcer (PPU) Non-specific Abdominal Pain (NAP), Cholisistitis (CHO), Intestinal Obstruction (INO), Pancreatitis (PAN), Renal Colic (RCO) and Dyspepsia (DYS). The category "Non-specific Abdominal Pain" is the term applied when no apparent cause was found for the patient's pain. The 33 most common symptoms with their different states are considered and there are 135 states of symptoms. We will use S_i (i=1,2, ..., 33) to denote the i-th symptom. For details, see Appendix D.

---

† The data were used with permission of Mr.S.J.Nixon of the Western General Hospital, Edinburgh and Mr.A.A.Gunn formerly of the Bangour Hospital, Roxburgh, where the data were originally collected.

5.1.3. Experiments on the Medical Database

We consider each symptom as a variable with finite states. In the database, each patient has one and only one disease. So these nine diseases are regarded as the different states of a disease variable D, acute abdominal pain. In total, there are 34 variables in our abdominal pain problem. For demonstration purposes, we also assume that all symptoms S_i are conditionally independent given the presence of a particular disease D_j. The graphical representation of our abdominal pain problem is presented in Figure 5.1. Furthermore, we assume that the probability distributions of variables in the graph have Dirichlet distributions. The parameters of the distributions can be estimated from the database.

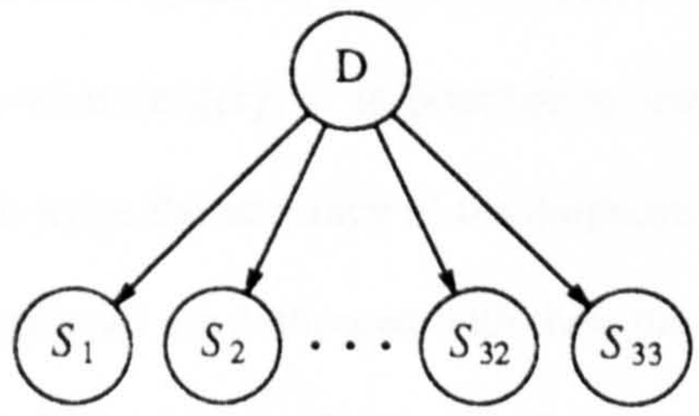


Figure 5.1. Graphical Representation of Abdominal Pain Problem

The database is randomly divided into a training and testing set. The training set is used to estimate all parameters of probability distributions needed to specify the above graphical representation. For example, the probability of a patient having symptom S_i given the presence of a disease D_j is specified by the parameter $\varepsilon_{(S_i, D_j)}$. The parameter $\varepsilon_{(S_i, D_j)}$ is estimated from the training set initially, for example,

$$\varepsilon_{(S_i, D_j)} = No.\ of\ patients\ with\ (S_i, D_j)\ in\ the\ training\ set\ .$$

When we fully specify the model from the training set, the remaining set (testing set) is used as new cases appearing in the hospital with abdominal pain for testing in order for the computer to make a diagnosis we exploit evidence propagation. PRESS developed in chapter 3 will serve as a computational framework for making diagnoses and updating the parameters [33].

When a patient record is taken from the testing set, the symptoms of the patient have been observed and propagated in PRESS. Our belief in the disease variable D will be revised in the light of the symptoms observed. Computer diagnoses are then made. The higher the probability of a certain diagnostic group, the more probable the patient would be diagnosed to have this disease. Consequently, according to this rule, the diagnostic group with highest probability is selected as the computer diagnosis for this new patient.

In order to evaluate the sequential learning technique, two systems, named System 1 and System 2, were implemented based on PRESS to carry out the computer diagnoses. In System 1, all the conditional probabilities needed were estimated from the training set and considered as precisely defined. However, System 2 is extended to allow us to specify all the conditional probabilities as Dirichlet distributions and update these parameters of the distributions in the light of new cases. In other words, it is able to learn from new cases in the testing set.

5.1.4. Evaluation

It is always important to evaluate a system's performance after it has been constructed. Since most of the patients in the database underwent surgery, it is possible to use the final diagnoses (the confirmed diagnoses of the cases) as a base to judge the accuracy of the diagnostic performance of our system. Given the testing set, all the computer diagnoses are compared with their final diagnoses made by the doctors and evaluation matrices are then built. Table 5.1 and Table 5.2 illustrate such matrices. The numbers on the diagonal line indicate the correct cases of computer diagnoses within different diagnostic groups. The overall accuracy can be worked out by the sum of these numbers divided by the total number of patient records in the testing set, as shown.

System 1 and System 2 are initialised on the same training set and tested with the same remaining (testing) set. For demonstration purposes, the training size is chosen as one of 100, 200, 300, 400, 500, 1000, 2000, 3000, 4000, 5000 respectively. The remaining set is used as the testing set. That is, the testing set is 5387 if the size of the training set is 1000. Matrices for comparison between final diagnoses and computer diagnoses are made for the different testing sizes. The matrices are given in Appendix E. For example, Table 5.1 and Table 5.2 give two matrices for comparisons between final diagnoses and computer diagnoses of System 1 and System 2 on the same testing data set (5,887 patient records). The accuracy is worked out by the sum of the numbers on the diagonal line divided by the size of the testing set, as previously. Three tests were made on the same size of the different training sets. The overall accuracy of computer diagnoses on different sizes of training sets is summarised in Table 5.3.

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	400	10	0	338	3	13	0	3	23	790
DIV	9	5	0	77	0	29	0	4	8	132
PPU	12	5	4	15	9	34	0	2	39	120
NAP	259	14	1	2059	38	59	1	58	134	2623
CHO	21	2	1	37	328	24	4	1	109	527
INO	29	20	0	112	17	172	0	1	35	386
PAN	4	1	1	8	17	9	3	0	46	89
RCO	27	2	0	104	19	7	0	245	24	428
DYS	22	2	1	88	59	20	3	3	594	792
Total	783	61	8	2838	490	367	11	317	1012	5887

Overall Accuracy=3810/5887=64.72%

Table 5.1. Comparison between Final Diagnoses and the Computer Diagnoses (System 1)

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	550	7	13	177	2	17	3	3	19	790
DIV	3	65	4	29	0	22	1	5	3	132
PPU	5	3	65	4	11	8	8	3	12	120
NAP	321	51	8	1946	42	83	2	76	96	2623
CHO	2	2	13	22	353	25	10	5	96	527
INO	11	22	10	46	10	247	7	5	25	386
PAN	2	1	6	4	14	9	19	1	34	89
RCO	9	12	3	59	10	6	1	314	13	428
DYS	12	7	14	64	60	20	19	7	589	792
Total	915	170	136	2351	502	437	70	419	887	5887

Overall Accuracy=4148/5887=70.46%

Table 5.2. Comparison between Final Diagnoses and the Computer Diagnoses (System 2)

Training Size	Overall Accuracy(%)	
	System 1	System 2
100	39.39	69.07
200	53.58	69.75
300	60.34	69.95
400	64.13	70.19
500	65.25	70.31
1000	69.25	70.89
2000	70.59	70.83
3000	71.14	71.28
4000	71.40	71.75
5000	71.81	71.79

Table 5.3. Overall Accuracy of Two Systems

To compare the overall accuracy of the two systems, a figure is drawn based on Table 5.3, see Figure 5.2. It is easily seen that the performance of system 2 is much better than that of system 1 when the size of the training set is small (e.g. <2000). For example, the accuracy of System 2 was near 70% when the size of training set is 300, but the accuracy of System 1 was about 60%. When the size of the training set is large, the performance of the two systems are almost identical. Given that the training set is 4000, System 1 achieved 71.4% of correct computer diagnoses and it is found that 71.7% of the computer diagnoses by System 2 were correct. The performance of the two systems (71%) is compatible with that of human doctors (76%). From these experiments, we conclude, not surprisingly, the performance of System 2 is more stable than that of System 1 as System 1 relies strongly on prior knowledge.

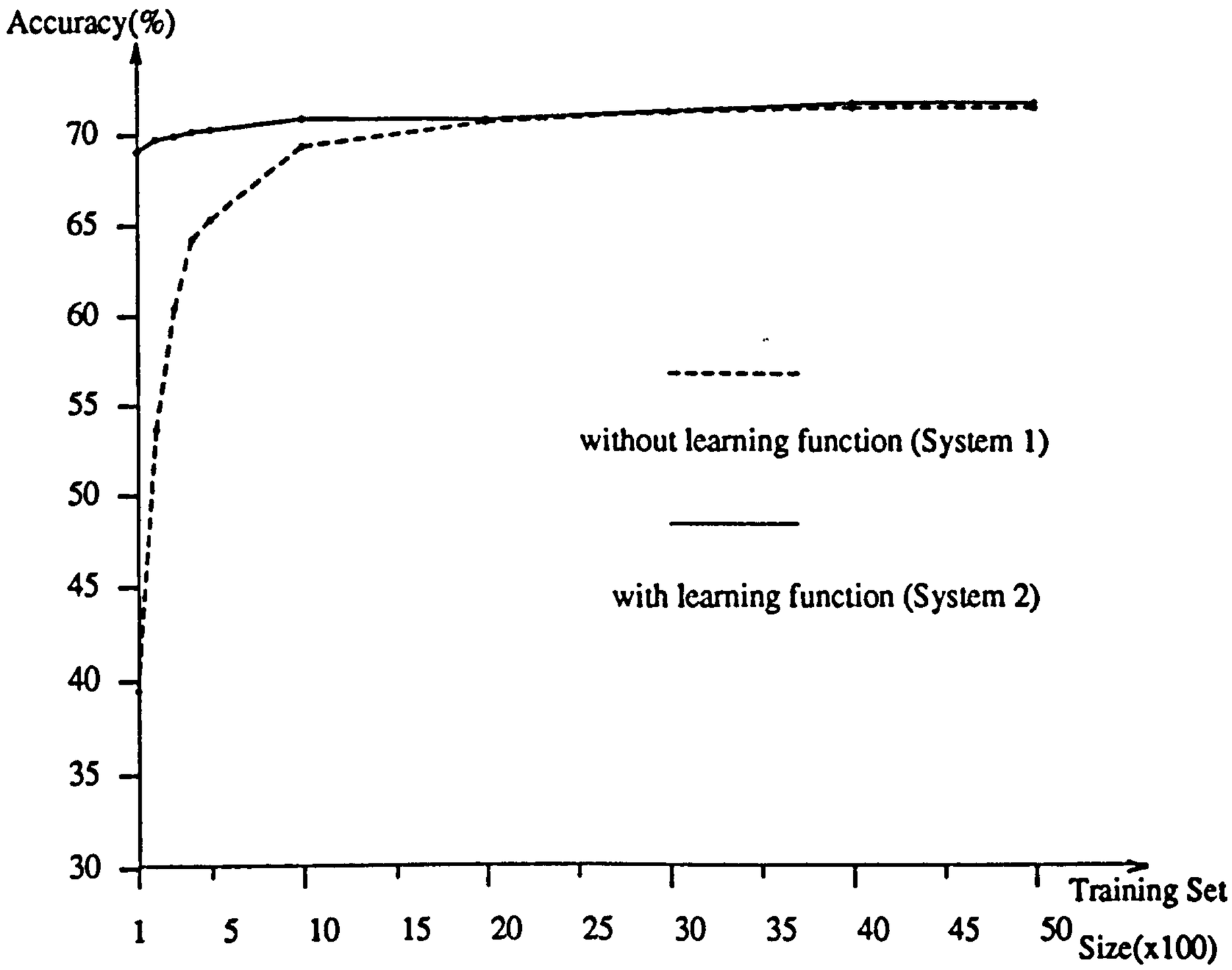


Figure 5.2. Comparison of Overall Accuracy of Two Systems

5.2. Structure Learning

In general forming a computational model based on Bayesian belief networks involves two distinct stages [122,123]. The first is the *construction of model* about some area of knowledge in the form of a

Bayesian belief network. The second stage involves the development of an *evidence propagation* procedure which will deal with any individual case and will allow the update of information on the graph in the light of new evidence.

The first stage may additionally have two different procedures to form the model. We can construct the model through a *knowledge elicitation* procedure through which an expert or a user may supply information about some area of knowledge in the form of a Bayesian belief network and associated conditional probabilities. Alternatively (or in addition) we may like to form the model in a semi-automated process by extracting the underlying topology directly out of the data, that is, using structure learning techniques. In this section, we will focus on learning structure rather than parameters, although obviously it is also needed to do some parameter estimation in order to produce a complete Bayesian belief network.

Given a database that consists of observations and attributes with corresponding values, the task of structure learning is to obtain a structure that explicitly captures as much information regarding conditional independencies as possible. The structure of interest here is represented in a graphical form - Bayesian belief networks. The structure has to be sparse so that it is comprehensible to the user and inference using the network is computationally tractable. There are two steps in structure learning:

(1) Extract a dependency model from data or expert's information. In other words, we try to reconstruct a probability distribution $\hat{P}(X_1, ..., X_n)$ of a dependency model.

(2) Construct a Bayesian belief network from the dependency model, that is, imposing directions on the model so that the joint probability probability distribution $\hat{P}(X_1, ..., X_n)$ can be expressed as

$$\hat{P}(X_1, ..., X_n) = \prod_{i=1}^{n} \hat{P}(X_i \mid PA(X_i)),$$

where $PA(X_i)$ is a set of parent nodes of X_i in the network.

In the rest of the section, two systems developed for structure learning are reviewed. The first system is called Kutato, an entropy-driven system for constructing Bayesian belief networks from databases [124]. The second system is developed by Srinivas, Russell and Agogino, a system for construction of Bayesian belief networks from dependence models specified by domain experts [125].

Kutato has two basic modules: one is to calculate the entropy of the joint probability distribution on a Bayesian belief network. The other is to construct a new Bayesian belief network based on entropy calculations and maximum-entropy principle. The conditional independence assumptions reflected in the Bayesian belief network are exploited and the entropy calculations simplified. During the network construction, Kutato begins with the assumption of marginal independence among all variables and adds the arc that maintains acyclicity and results in a Bayesian belief network with minimal entropy. We attempt to minimise entropy since we are approaching the maximum-entropy distribution. This is because the maximum-entropy principle states, that in the absence of prior information about the distribution, by choosing the full joint probability distribution that has the maximum entropy given the information at hand, we guarantee that probabilities derived from the resulting distribution will have no bias. As an arc is added, the database is used to update the conditional probability distribution. A new entropy is calculated. Arcs are added in this manner until a threshold is reached in the rate of decrease of the entropy between two successive networks.

However, it is assumed that directions of arcs are given by a domain expert, although arc directions could be determined based on the maximum entropy principle. Kutato has been tested on two artificial databases generated by a probabilistic logic sampling method. The behaviour of Kutato in real applications is not clear.

Srinivas, Russell and Agogino [125] developed an algorithm that takes as input some qualitative information from an expert about the dependence in the domain and returns a belief network incorporating these constraints. The information given by experts about a variable or set of variables may be expressed in any one of the following forms:

- a variable is a hypothesis variable
- a variable is an evidence variable
- A causes B
- X is conditionally independent of Y given Z, where X, Y and Z are sets of variables.

They assume that it is usually easy for an expert to identify these "primary" dependencies. This information is the basis for building a Bayesian belief network. For example, a hypothesis variable is a root node and an evidence variable is a leaf node in the Bayesian belief network. Cause-effect relationships are

interpreted as parent-child representations. The network is built incrementally adding one node at a time. The algorithm applies a priority heuristic to each node, adding root nodes, parents, children and leaf nodes to the belief network in that order and tries to keep the number of arcs added at each step to a minimum. Expert information is used merely to guide the search for the next node X_i. Ties are broken by adding the node that would bring with it the fewest arcs. The boundary stratum B_i of the next node X_i is found and those nodes in B_i are parents of node X_i. The boundary stratum B_i of node X_i is defined as a minimal subset of $\{X_1, ..., X_{i-1}\}$ such that X_i is conditionally independent of $\{X_1, ..., X_{i-1}\}-B_i$ given B_i. This process continues until all nodes have been added to the Bayesian belief network.

However, the algorithm's computational complexity is exponential in the number of nodes. It does not use data and constructs a Bayesian belief network from a dependence model specified by a domain expert. Therefore it only addresses the second task of structure learning.

5.2.1. Polytree Algorithm

The difficulty of Bayesian belief network discovery is apparent when one considers the number of possible models for a given set of variables. For 12 variables, the number of possible directed acyclic graphs is approximately 5.2×10^{26} [126]. In this section, we will focus on a particular kind of Bayesian belief networks, polytree structures, and polytree algorithm [29].

The polytree algorithm proposed by J. Pearl [29] consists of two steps: the first is to construct a skeleton tree and the second is to identify the direction of the branches of the tree by using a series of tests for independence - statistical independence tests on data.

5.2.1.1. Skeleton Tree

In the first step the main idea is to use a notion of tree dependence to approximate the underlying probability distribution data. In particular, the algorithm allows us to find the best approximation of an n-order distribution by a product of (n-1) second order distributions. The basic idea was developed by Chow and Liu [127] and the main result can be formulated as follows.

A probability distribution is called a distribution of the tree dependence if it has the following form:

$$P_t(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid X_j) \qquad j \in (0, 1, ..., n) \text{ and } i \neq j ,$$

where $P(X_i \mid X_0)$ is by definition equal to $P(X_i)$. A probability distribution of a tree dependence $P_t(X_1, X_2, ..., X_n)$ is an optimum approximation to a "real" distribution $P(X_1, X_2, ..., X_n)$ if and only if its dependence tree t has maximum weight, where the weight is determined by the mutual information measure $I(X_i, X_j)$ between two linked variables X_i and X_j:

$$I(X_i, X_j) = \sum_{X_i, X_j} P(X_i, X_j) \log \frac{P(X_i, X_j)}{P(X_i)\, P(X_j)} \geq 0 . \tag{5.2}$$

It has been proved by Chow and Liu that maximising the total branch weight is equivalent to minimising the the Kullback-Liebler measure [127]:

$$D(P, P_t) = \sum_{X} P(X_1, X_2, ..., X_n) \log \frac{P(X_1, X_2, ..., X_n)}{P_t(X_1, X_2, ..., X_n)} . \tag{5.3}$$

This measure can be interpreted as the difference between two distributions: it is always positive when the distributions are different and is zero when they are identical. Here is the algorithm for constructing the maximum weight spanning tree (MWST):

```
FOR i=1 to n-1 DO
  FOR j=i+1 to n DO
   BEGIN
      find out all second-order probability distributions P(Xi, Xj)
      calculate mutual information measure I(Xi, Xj) by the expression (5.2)
   END
Branches_No=0
WHILE Branches_No < (n-1) DO
BEGIN
     select two variables Xi, Xj who have largest I(Xi, Xj)
     add the branch (Xi, Xj) to the tree
     IF there is a loop in the tree
     THEN delete the branch (Xi, Xj)
     ELSE
     BEGIN
          Branches_No=Branches_No+1
          print out the branch (Xi, Xj)
     END
END
```

Algorithm 1: Constructing a Maximum Weight Spanning Tree

There are $n(n-1)/2$ pairs of $I(X_i, X_j)$, and the algorithm terminates when $(n-1)$ branches have been selected, at which point, the dependency tree has been constructed. The time complexity of Algorithm 1 is $O(n^2)$.

5.2.1.2. Recovering Directionality

Having constructed the skeleton tree the next task is to recover the directions of the branches. According to the polytree algorithm we consider each internal node (the node that has more than one neighbour), and apply an independence test. That is if, say, X_k is an internal node and has at least two neighbours X_i and X_j, then we try to establish whether X_i and X_j are marginally independent. If they are, then we assign directions from X_i to X_k and from X_j to X_k, see Figure 5.3.

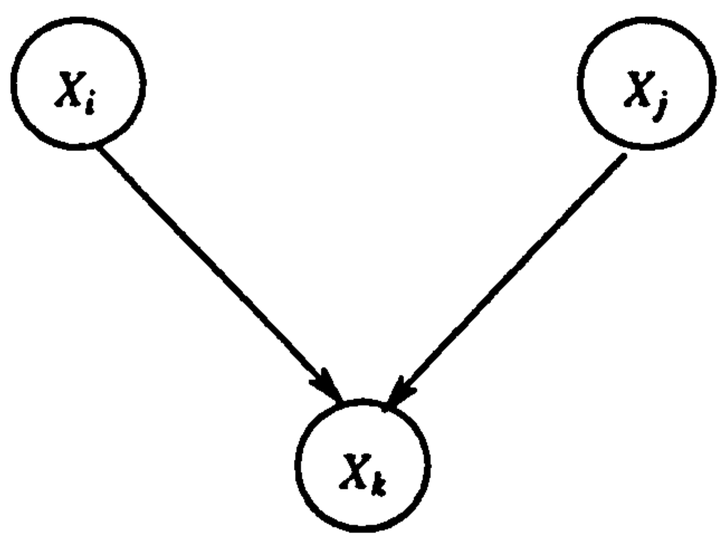


Figure 5.3. Marginal Independence

If, however, X_i and X_j are not marginally independent then there are two possible assignments: from X_k to X_i and X_j or from X_k to X_i and X_j to X_k, see Figure 5.4. More information is needed to recover the directions in this case.

Figure 5.4. Possible Conditional Independence

We repeat this procedure for all possible combinations of pairs for each internal node, and assign directions when appropriate. Some assignments, however, may be inconsistent (the arrows in a branch may go in both directions) due to the different types of dependency structures. As it has been shown, only one type of dependency can be uniquely identified [29], therefore, only a "partial" recovery may be possible.

As an independence test, we can continue to use the mutual information measure. This time we can exploit the fact that if X_i and X_j are independent then the measure 2NI, where N is the number of cases in the database and I is the mutual information measure between X_i and X_j, is asymptotically (as $N\rightarrow\infty$) distributed as central χ^2 with (r-1)(c-1) degrees of freedom, where r and c are the number of values of variables X_i and X_j respectively [103]. For computational convenience we use the following formula:

$$\hat{I} = \sum_i \sum_j F_{ij} \log F_{ij} - \sum_i F_i \log F_i - \sum_j F_j \log F_j + N \log N = NI \ , \qquad (5.4)$$

where F_{ij}, F_i and F_j are the frequencies of occurrencies in the database.

The algorithm for identifying direction is summarised as follows:

```
FOR i=1 to n DO
BEGIN
    IF  X_i has more than one neighbour
    THEN put X_i in MULTIPLE_SET
END
FOR each X_i in MULTIPLE_SET DO
BEGIN
    FOR any pair of neighbours (X_j , X_k) of X_i DO
    BEGIN
        IF X_j and X_k are independent
        THEN
        BEGIN
            2Î is distributed as χ² with (r-1)(c-1) degrees of freedom
            (where Î is the mutual information measure calculated by the expression (5.4))
            X_j → X_i , X_k → X_i
        END
        ELSE
        BEGIN
            IF X_j → X_i THEN X_i → X_k
            IF X_k → X_i THEN X_i → X_j
        END
    END
END
```

Algorithm 2: Identifying Direction

Note that the algorithm we used is slighdy different from the one described in [29]. In our calculations we only check whether a node is an internal node, and if it is we immediately apply the independency test. In Pearl's setting the algorithm also checks whether the internal node is a multiple parents node. The algorithm then checks whether pairs have partial recovered connections and carries out the independency test.

5.2.2. Experiments

Two database are used for experiment purposes. One is a simulated database. The other is the medical database. The brief description of the simulated databases is given below.

5.2.2.1. A Simulated Database

The simulated database is generated by the stochastic simulation algorithm described in section 2.5.1 and used for the experiments of structure learning. The Bayesian belief network employed is shown in Figure 5.5, where the numbers are the labels for variables. As a fairly simple example, there are 10 variables and all of them are assumed to be binary. The corresponding conditional probabilities are given in Table 5.4.

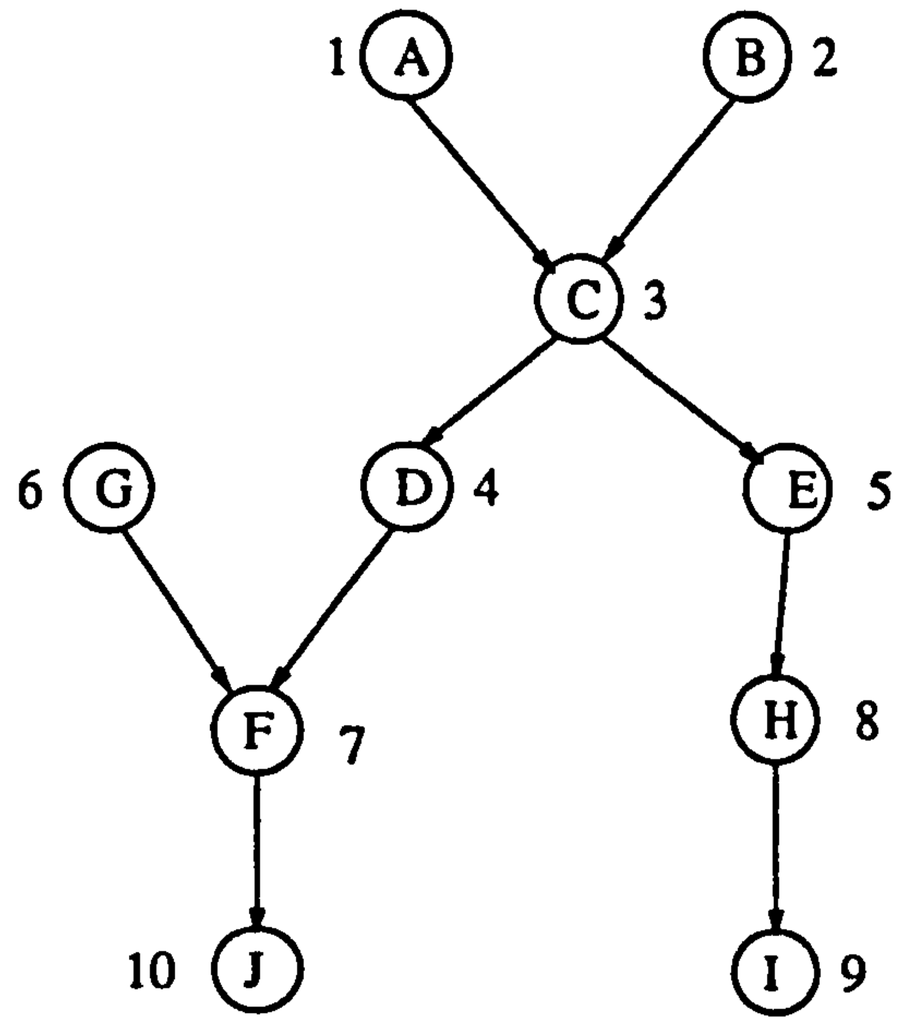


Figure 5.5. Bayesian Belief Network

P(a)=0.4	P(b)=0.8	P(g)=0.6
P(c|a,b)=0.3	P(f|d,g)=0.9	P(d|c)=0.8
P(c|a,¬b)=0.9	P(f|d,¬g)=0.3	P(d|¬c)=0.4
P(c|¬a,b)=0.8	P(f|¬d,g)=0.4	P(e|c)=0.3
P(c|¬a,¬b)=0.6	P(f|¬d,¬g)=0.3	P(e|¬c)=0.5
P(h|e)=0.4	P(i|h)=0.4	P(j|f)=0.7
P(h|¬e)=0.5	P(i|¬h)=0.6	P(j|¬f)=0.4

Table 5.4. Conditional Probability Table

5.2.2.2. Experiments on the Simulated Database

A database of 1000 examples is simulated. Based on the database, the estimated information measure is given in the following table.

Order	Selected Branches	Information Measure
1	(3, 4)	0.102626
2	(6, 7)	0.071749
3	(1, 3)	0.054858
4	(4, 7)	0.046570
5	(7,10)	0.038678
6	(3, 5)	0.016773
7	(8, 9)	0.012174
8	(1, 4)	0.011094
9	(2, 3)	0.009437
10	(5, 8)	0.004981

The skeleton tree constructed is the same as the original tree, see Figure 5.5. The independence test is then carried out on the skeleton tree. The χ^2 values are calculated. These values are given in the following table

Internal Node	Neighbour Pairs	Degrees of Freedom	χ^2
3	(1, 2)	1	0.021788
	(1, 4)	1	22.188168
	(1, 5)	1	6.280303
	(2, 4)	1	0.921063
	(2, 5)	1	0.033933
	(4, 5)	1	14.511055
4	(3, 7)	1	14.902133
5	(3, 8)	1	1.501626
7	(4, 6)	1	3.718690
	(4,10)	1	6.825584
	(6,10)	1	13.982370
8	(5, 9)	1	8.830268

The hypothesis of independence is accepted or rejected by comparison of χ^2 statistic. We choose the χ^2 value at the 5% level of significance, that is, $\chi_{.95}^2(1)=3.84$. The directions on the tree is depicted in Figure 5.6. Note that links (2,4), (2,5) and (3,9) have both directions. In other words, their directions are not uniquely determined.

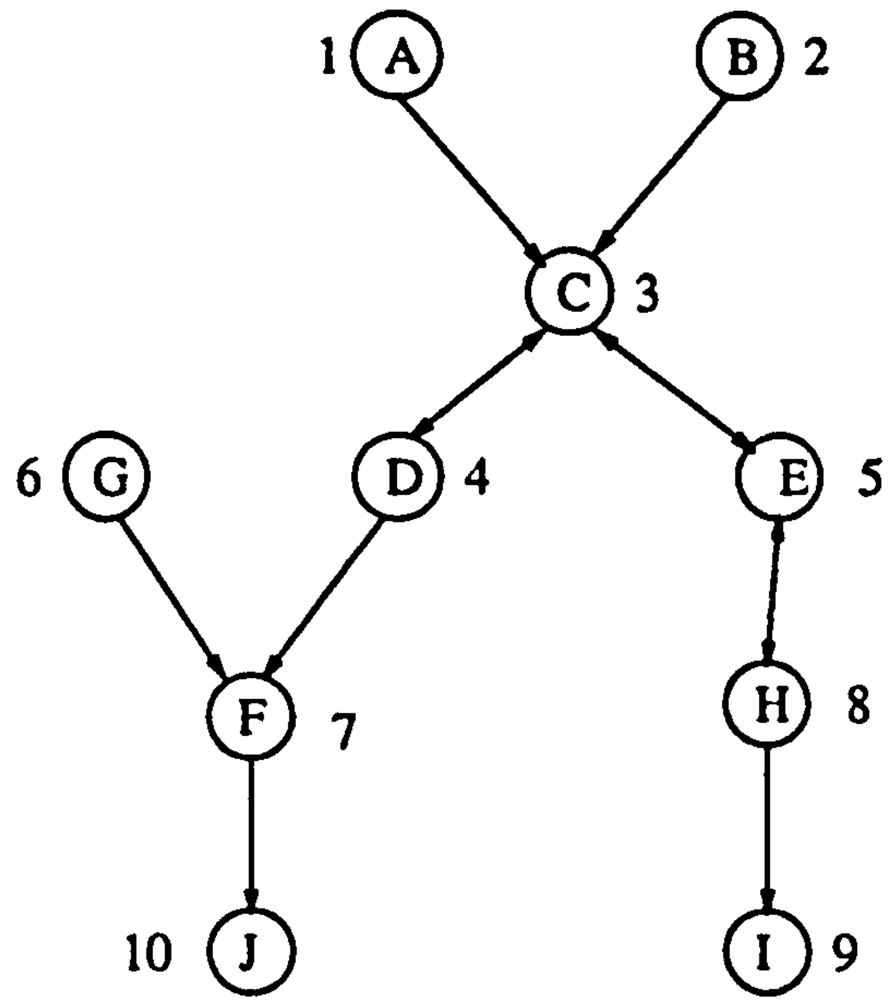


Figure 5.6. Reconstructed Bayesian Belief Network

5.2.2.3. Experiments on the Medical Database

Skeleton Tree

In order to apply the Chow-Liu algorithm we consider each symptom with corresponding values as a variable (and a node in the tree). The 9 diagnostic groups are represented as a disease node with 9 values. So there are 34 variables in total in the skeleton tree. For brevity we use reference numbers (1 to 33) to denote symptoms, and No. 34 to denote the disease variable.

The database is randomly divided into 2 sets: the training set of 4,387 patient's records and the testing set of 2,000 records. The training set is used to construct a polytree and estimate all prior and conditional probabilities. The testing set is used to evaluate the polytree.

According to the Chow-Liu algorithm the information measures between all pairs of variables have been calculated and the maximum weight spanning tree (MWST) has been constructed by ordering the

measure and choosing only those branches that did not form a loop. The resulting table for 33 branches is given below, see Table 5.5.

Order	Selected Branches	Information Measure
1	(3, 4)	0.968999
2	(4,26)	0.752362
3	(26,34)	0.466640
4	(19,24)	0.407164
5	(2,34)	0.253484
6	(5, 6)	0.162493
7	(21,23)	0.136150
8	(26,33)	0.097465
9	(21,22)	0.089500
10	(5,34)	0.088761
11	(31,34)	0.086216
12	(27,34)	0.085424
13	(2,19)	0.082851
14	(22,34)	0.082084
15	(32,34)	0.078212
16	(9,24)	0.076851
17	(11,12)	0.070648
18	(14,34)	0.067856
19	(27,28)	0.067694
20	(25,34)	0.067312
21	(2,20)	0.064471
22	(10,21)	0.059096
23	(4 ,7)	0.058322
24	(17,34)	0.057519
25	(23,29)	0.049169
26	(2,16)	0.046241
27	(8,34)	0.046079
28	(11,13)	0.044869
29	(1,34)	0.038894
30	(12,34)	0.036183
31	(15,22)	0.031611
32	(18,34)	0.031181
33	(30,34)	0.019067

Table 5.5. Branches of the Skeleton Tree

Figure 5.7 shows a MWST constructed on the base of those results, where the node No. 34 (diagnostic node) has been chosen as a root of the tree.

Directions on the MWST

Using the test for independence described above we can assign directions to the skeleton tree (see Appendix F). However, there are some problems which arise in assigning directions to the MWST. For example, some directions are clearly meaningless from medical point of view. In order to avoid this a medical expert's advice has been sought and as a result of the consultation the skeleton tree has been

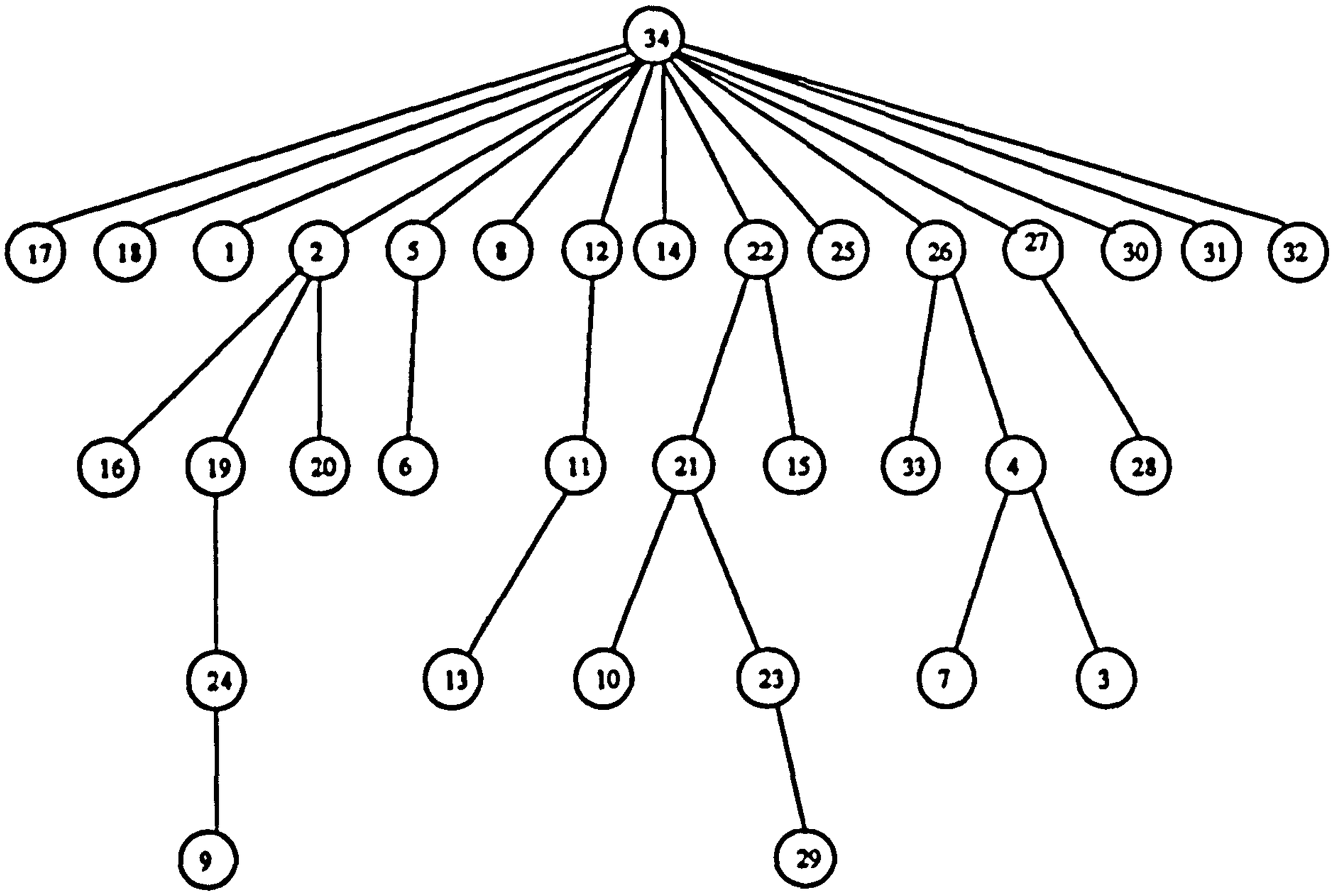


Figure 5.7. Skeleton Tree

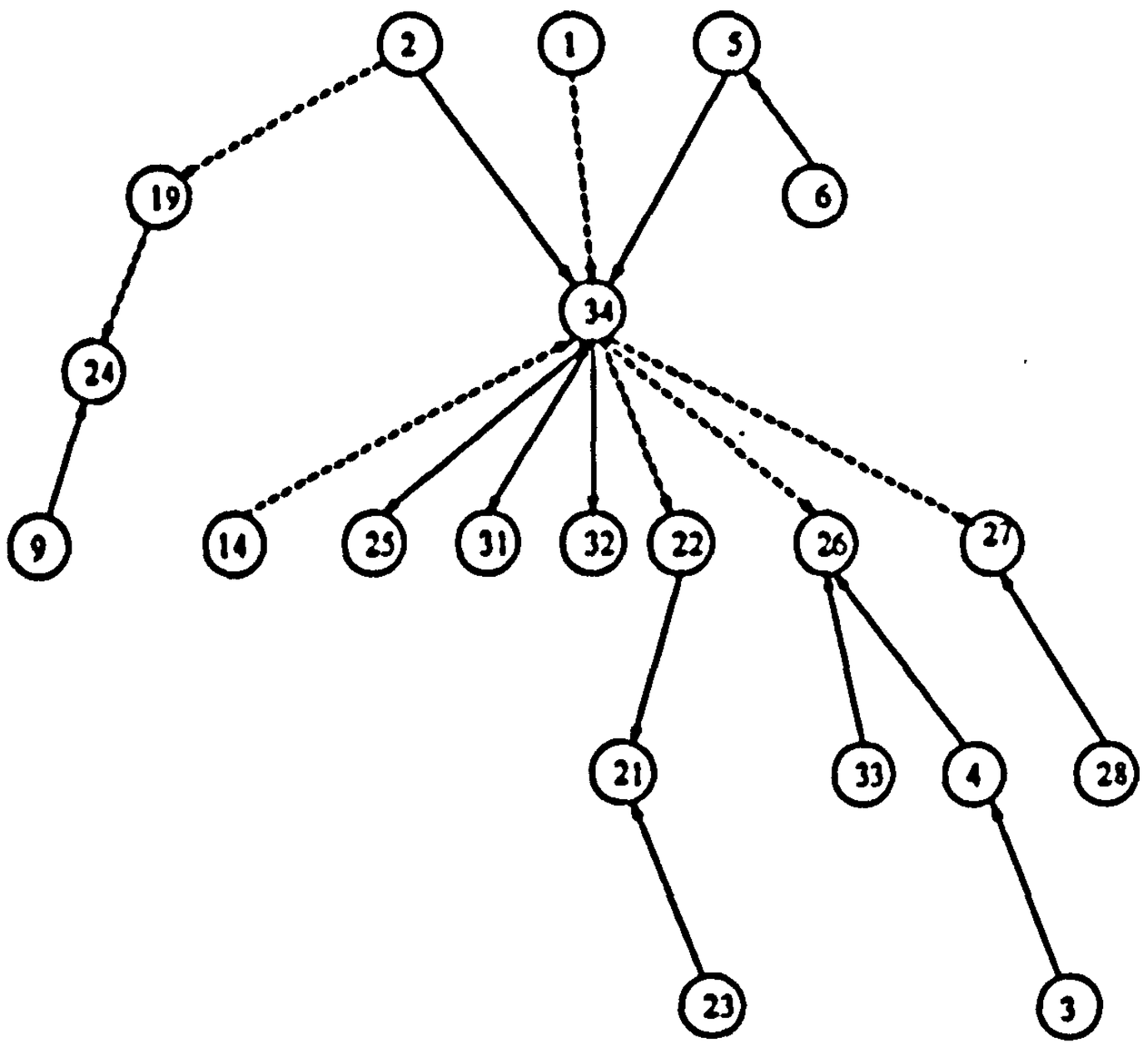


Figure 5.8. Pruned Polytree

pruned by choosing a threshold (I=0.065). After that the test for independence for all pairs of variables X_i and X_j of each internal node has been applied. That is when the 0.005% values of χ^2 with respective

degrees of freedom is less than the doubled mutual information measure $2\hat{I}$, then we accept the hypothesis of independence between variable X_i and X_j. In some cases where both directions can be assigned to the same branch, we did not assign the directions at all. For these branches where the algorithm could not assign any directions we use the doctor's advice and impose the directions (dashed lines) suggested by them. The corresponding pruned polytree is represented in Figure 5.8.

After constructing the tree we can use our training set in order to supply all necessary prior and conditional probabilities. At this stage when we obtained a qualitative structure in the form of polytree, and quantitative information concerning this tree, the *model construction* phase is finished. Now in order to make the computer diagnosis we can exploit the *evidence propagation* procedure in PRESS.

Computer Diagnoses

Having constructed a polytree with associated probabilities we can use one of the propagation techniques developed in PRESS in order to make a diagnosis†. That is we propagate observations of symptoms of a new patient through this tree and re-assess the probabilities of the diagnostic node. The node has 9 values and shall have 9 probabilities for each patient from the testing set. As a "computer diagnosis" then we take the highest probability and interpret it as a diagnosis for this patient. Having a sample of 2,000 patients we can evaluate our accuracy by comparing the results with final diagnoses given by doctors, see Table 5.6. About 68 per cent of these diagnoses are correct in the sense that they agree with the final diagnoses.

5.2.3. Evaluation

It would be interesting to compare the performance of the model constructed by the polytree algorithm with the statistical methods developed to help knowledge engineers to construct a knowledge base from data. Here we consider the following methods: the ID3 algorithm in the C4 system [53,54], the method used in the CART (Classification And Regression Trees) system [55], the G&T method described

---

† Note that it can also be done directly from the skeleton tree by estimating unknown joint probability distributions. The results are shown in Appendix F.

Computer Diagnoses vs. Final Diagnoses										
	Computer Diagnoses									
Final Diagnoses	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	178	1	3	59	0	9	2	0	7	259
DIV	2	14	0	6	0	9	0	0	4	35
PPU	2	1	19	2	3	6	6	0	3	42
NAP	106	20	3	683	11	21	0	16	34	894
CHO	2	0	4	7	128	14	4	4	37	200
INO	8	14	4	22	6	59	1	2	11	127
PAN	0	0	2	1	6	1	8	0	13	31
RCO	3	3	0	27	6	12	2	92	2	147
DYS	4	1	7	27	22	4	5	4	191	265
Total	305	54	42	834	182	135	28	118	302	2000

Overall Accuracy=1372/2000=68.6%

Table 5.6. Comparison between Computer Diagnoses and Final Diagnoses

Computer Diagnoses vs. Final Diagnoses										
	Computer Diagnoses									
Final Diagnoses	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	120	0	6	119	3	5	0	0	6	259
DIV	0	9	2	19	1	2	0	1	1	35
PPU	2	0	19	12	2	0	0	0	7	42
NAP	48	9	0	758	18	13	0	24	24	894
CHO	2	1	1	42	121	5	0	4	24	200
INO	1	6	0	53	2	54	0	0	11	127
PAN	0	0	2	14	3	2	1	2	7	31
RCO	2	2	0	69	1	0	0	73	0	147
DYS	1	0	3	85	21	4	0	5	146	265
Total	176	27	33	1171	172	85	1	109	226	2000

Overall Accuracy=1301/2000=65.0%

Table 5.7. Performance of G&T System (with Threshold 11.5)

in section 2.6 and the Simple Bayes approach. Note that ID3 algorithm and CART method are modified and implemented under the G&T system structure [22]. In the Simple Bayes approach an assumption of independence is used among all the considered attributes of a case to reduce its calculation complexity. These methods have been applied to the same data set, a training group consisted of 4387 patients for extracting knowledge in helping with later decision making and a testing group made up of the remaining 2000 for evaluating each of the approaches. For the CART and the ID3 approaches, the 4387 patient records were used for constructing a decision tree, using different termination conditions. For the G&T approach, the best accuracy 65.0% happens when setting the threshold to 11.5, see Table 5.7. We found

that the accuracy the CART approach could reach is 64.6% (shown in Table 5.8) when α=0.015 is set. Similarly, an accuracy of 65.0% was achieved by the ID3 algorithm when setting 3 for attribute suitability, 70% for class frequency and 0.8% for node weight (Table 5.9). When the Simple Bayes approach was applied, an accuracy of 73.8% is reached and the detailed matrix is shown in Table 5.10. Their overall accuracies are summarised in Table 5.11 together with that of human doctors diagnoses. The doctors perform diagnosis more accurately than computer programs. Apart from the Simple Bayes approach, the best accuracy achieved from the other three approaches is quite similar. Furthermore, they are at the same level of accuracy as that of using structure learning technique (68.6%) although the success rate of using structure learning is slightly higher. This is because that the Simple Bayes approach is over-confident in dealing with probabilities. Simple Bayes model needs less parameters. The complexity of the Simple Bayes model is low. Therefore, the performance of the Simple Bayes approach is expected to be better. The database may not be sufficiently large for all the relevant combinations of symptoms to be identified in adequate detail [22].

Computer Diagnoses vs. Final Diagnoses										
	Computer Diagnoses									
Final Diagnoses	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	91	1	10	141	7	3	0	0	6	259
DIV	0	0	3	23	1	6	0	1	1	35
PPU	1	2	21	6	3	5	0	0	4	42
NAP	32	8	3	744	32	14	0	19	42	894
CHO	0	6	4	24	136	10	0	2	18	200
INO	2	2	7	39	1	64	0	1	11	127
PAN	0	1	4	5	5	2	0	2	12	31
RCO	1	2	0	64	4	2	0	74	0	147
DYS	0	4	6	53	34	3	0	2	163	265
Total	127	26	58	1099	223	109	0	101	257	2000

Overall Accuracy=1293/2000=64.6%

Table 5.8. Performance of CART System (α=0.015)

5.3. Discussion

The sequential learning algorithm suggested by Spiegelhalter and Lauritzen [36] gives us an opportunity to investigate parameter learning. The experiments on the medical database show that it is easy to represent and estimate parameters from a database when the distribution is Dirichlet. Sequential learning

Computer Diagnoses vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	97	3	1	144	0	7	0	0	7	259
DIV	0	3	1	18	0	4	0	4	5	35
PPU	1	3	15	6	3	4	0	2	8	42
NAP	41	10	0	758	12	19	0	20	34	894
CHO	0	2	0	32	122	11	0	8	25	200
INO	1	7	0	41	2	62	0	1	13	127
PAN	0	2	3	5	7	3	0	3	8	31
RCO	1	3	0	57	3	1	0	80	2	147
DYS	1	4	4	55	30	5	0	3	163	265
Total	142	37	24	1116	179	116	0	121	265	2000

Overall Accuracy=1300/2000=65.0%

Table 5.9. Performance of ID3 System

Computer Diagnoses vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	192	1	5	47	0	4	2	0	8	259
DIV	1	22	3	3	0	6	0	0	0	35
PPU	0	1	32	2	2	0	3	1	1	42
NAP	106	18	4	672	16	21	0	32	25	894
CHO	2	1	6	5	152	9	6	1	18	200
INO	3	11	2	9	2	92	2	1	5	127
PAN	0	0	4	0	6	1	7	2	11	31
RCO	4	3	1	18	2	0	1	116	2	147
DYS	2	2	7	21	26	8	6	1	192	265
Total	310	59	64	777	206	141	27	154	262	2000

Overall Accuracy=1477/2000=73.8%

Table 5.10. Performance of Simple Bayes System

Method	Correct Diagnoses (per cent)
ID3	65.0
CART	64.5
G&T	65.0
Simple Bayes	73.8
Structure Learning	68.6
Human Doctors	76.3

Table 5.11. Summary of Results of Different Methods

is an effective way to handle imprecision of conditional probabilities and adjust the probabilities in the model. The accuracy of the computer diagnosis system is improved considerably with the learning function

compared with that without a learning function, especially when the past data set is small. The overall performance of the learning system is stable and not that sensitive to the prior knowledge. The computation of the technique is quite flexible as only some extra calculations are needed to update the parameters.

The medical database has a complete set of observations and the updating is straightforward. It would be interesting to investigate the performance of the technique in the case of missing data. The method is at the cost of maintaining the network structure constant over time, and independence assumptions are made to achieve local operations. It restricts these techniques to be applied only to domains where the independence assumptions hold. The consequences of these independence assumptions are not clear.

In our experiments, the two-level graphical structure used for the parameter learning may not be appropriate. It does not reflect the "causal" structure. Perhaps some factors, like "alcohol consumption", could be added to the structure to form another level. It may be important to make the system forget the past at an exponential rate, thereby making it more prone to adapt in a changing environment.

In an effort to address the problem of structure learning, the polytree algorithm for network induction [29] has been explored. There are a number of problems associated with the polytree approach to the medical database. First of all, as has been said some of the directions on the tree are meaningless from a medical point of view. They certainly do not represent "causal" directions. The most we can say about some of them that they reflect "commonality". For example, the link from node 3 (Pain-site Onset) to node 4 (Pain-site Present), see Appendix F. That means that we should develop methods to distinguish "causal" from other types of associations. Temporal precedence is one of the most important clues that people can use. There is also the problem of distinguishing genuine causation from spurious associations caused by unknown factors. Secondly, some links have arrows going in both directions. For example, branch 11↔12 or 21↔22 (see Appendix F). Therefore, for these links the directions can only be assigned arbitrarily. This problem has been mentioned before and it is connected with non-unique recovery due to different types of dependencies. Thirdly, some nodes have far too many parent nodes. For example, the disease node has 9 parent nodes in our polytree. In order to specify the conditional probabilities for the node, more than 6 million probabilities are needed. It is unrealistic to propagate evidence using this amount of information.

The problem can be avoided by introducing a threshold, or to employ the idea developed in the G&T system [38]. In the latter case we need to estimate combinations of the variables which actually occur in the database. The design of such modification constitutes an important path for future research.

More generally, although this algorithm is relatively efficient computationally, it is highly restricted in that only those approximating distributions composed of second order probabilities are considered. This will effectively make the data fit into a tree (even the best tree is still a tree structure). Furthermore, there is no guarantee of finding a polytree for an arbitrary distribution. These structures are often not expressive enough to represent real-world situations. The assumption of a tree dependence distribution should be relaxed. A future research topic will be the generalisation of the polytree algorithm for networks. Asymptotically correct algorithms for recovering sparse graphs are expected to be developed.

An additional and more basic problem with statistical independence tests on data is that they can never be exact. Therefore, direction recovery is very subtle. The experiments on the simulated data in section 5.3.2.1 highlight this point. Methods should be developed to overcome this problem. In addition, it is difficult to distinguish genuine causal dependencies from spurious correlations among observable variables. From our experience with the medical database, we construct a Bayesian belief network from a database presented by the domain experts. The structure is then criticised and modified. We found it is a satisfactory way to combine effectively the expert's opinions with data. We believe the domain expert to be essential for structure learning, especially for assignment of directions. This belief is also put forward by [125].

It would be interesting to combine parameter learning and structure learning. We might also incorporate devices for monitoring the adequacy of a given model and/or of the prior distributions embodied in it to the problem at hand. If we consider a number of possible network structures, then we can monitor the predictive probability of the data obtained on each case which makes the global comparison simply the product of the Bayes factors obtained while treating each case [128,129]. Based on these measures, the learning task involves criticisms of the network structure and possibly modifying it dynamically. It would be a future research topic.

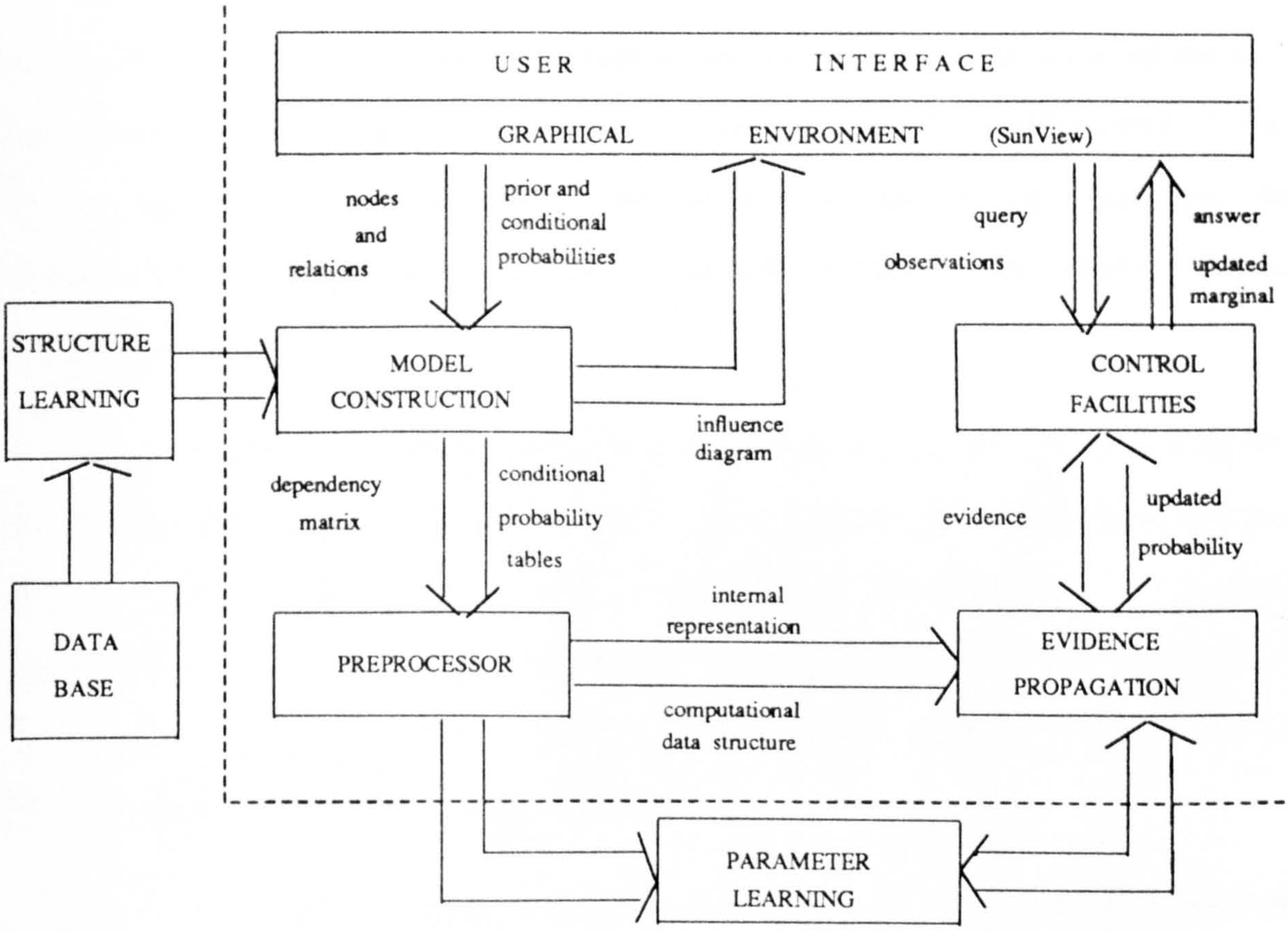


Figure 5.9. Architecture of the Integrated System PRESS

PRESS is extended to accommodate different learning techniques and provides a platform for experiments on learning in Bayesian belief networks. The architecture of Probabilistic Reasoning Expert System Shell (PRESS), consists of four main modules: Model Construction, Preprocessor, Enter Evidence and Evidence Propagation. Two additional modules, the Structure Learning and the Parameter Learning, are integrated into the system. The architecture of the integrated computational system is presented in Figure 5.9. The integrated system is written in C and implemented under SunOS. It runs on a SUN SPARC workstation SLC.

5.4. Concluding Remarks

When past data is available, Bayesian statistical techniques can be used for modifying the models and machine learning methods for constructing the models. Two related learning tasks in Bayesian belief networks, parameter learning and structure learning, are considered in this chapter. Structure learning is one of the most important, but also most controversial.

Based on a Bayesian belief network representation and some independence assumptions, the sequential learning technique proposed by Spiegelhalter and Lauritzen [36] can be performed locally to cope with imprecision of conditional probabilities. The set up makes it possible to take advantage of well-developed statistical techniques and evidence propagation algorithms based on local computations. The independence assumptions provide a computationally straightforward basis for local operations on these conditional probabilities [36].

We demonstrate how to apply sequential learning method to a special case, when probability distributions can be expressed as Dirichlet distributions. These parameters of the distribution are estimated from a accumulated database. The simplicity of operations on the distributions and its intuitive interpretation are very attractive. Experiments with the method have been applied to the medical database where computer diagnosis systems are developed based on PRESS. The results show a number of attractive properties of the sequential learning procedure.

On the other hand, constructing Bayesian belief networks by knowledge engineers working with domain experts who provide the entire information is a bottleneck in developing probabilistic expert systems. Structure learning provides an alternative way to bypass such difficulties and forms a model directly from a database.

The task of structure learning is to obtain a structure "Bayesian belief network" that captures explicitly as much information regarding conditional independences as possible given a quality database. The work conducted in this chapter is only one approach towards the issue with the assumption that underlying distributions can be approximated by a dependence tree. The polytree algorithm described by Pearl [29] has been developed and experimented on databases. The polytree constructed from the medical database is then used for predicating diagnosis based on PRESS when a new case is obtained. The performance is comparable with other methods at the level of overall accuracy.

The learning process integrates a "subjective expert's approach" with an "objective database approach" and greatly enchances PRESS. The application to the medical database provides an opportunity to gain experience. A number of problems associated with the learning process are identified and possible further developments suggested. The empirical investigations show that learning will allow the

development of "adaptive probabilistic reasoning systems" although more effort should be given to studying learning, especially structure learning, in Bayesian belief networks.

Chapter 6

CONCLUSIONS

In this concluding chapter, the underlying problems of probabilistic reasoning in knowledge based systems are reviewed. The approaches taken in this thesis are described and the computational framework that has been developed presented. Finally, future research in this area is suggested.

6.1. The Problems

As discussed in chapter 2, the early development of probabilistic expert systems had two kinds of difficulties. The first was how to construct and encode expertise into a coherent probabilistic form, namely a representational problem. In order to achieve consistency, it had to make ad hoc function adjustments due to the inappropriateness of rule-based systems for representing uncertainty [12]. The second was the inferential problem, that is, how to develop a computationally manageable algorithm for reasoning and decision making. In order to cope with the problem in practice, many Bayesian inference systems had to make some unrealistic assumptions, for example, the conditional independence assumptions in PROSPECTOR [2]. Some systems avoid independence assumptions but need a large database (e.g. G&T) [38].

In summary, to follow a normative probabilistic approach in uncertainty management, researchers had to make a choice between oversimplified models and computationally intractable domain models. This has led to the development of other uncertain reasoning mechanisms [13,15,16]. For example, the certainty factor model was developed because the MYCIN system needs a computationally feasible inference mechanism [1]. These approaches, which concentrate on specific aspects of uncertain inference, have given rise to inadequacies in other aspects.

6.2. The Approaches

Bayesian belief networks (BBNs), which were introduced in chapter 2, have the nice property that the joint probability distribution can be expressed as the product of corresponding prior and conditional probabilities specified on the graph [28,74]. These probabilities are guaranteed to define a coherent joint probability distribution on the graph. The representational problem is thus overcome to a great extent. By exploiting and taking full advantage of the independence relationships embodied in the Bayesian belief network, the computation of the joint probability distribution (global computation) on all variables can be decomposed into a set of computations on small groups of variables (local computations). These local computations can be carried out one by one. This also eases inferential difficulties arising in probabilistic reasoning and avoids making independence assumptions in general.

Three notable probabilistic reasoning algorithms using Bayesian belief networks, namely, Pearl's message passing algorithm [25], Lauritzen-Spiegelhalter's clique tree algorithm [28] and the stochastic simulation algorithm [32] were critically studied and carefully evaluated in chapter 2. Both Pearl's and the L-S algorithms are exact algorithms (with exact values of belief) [25,26,28]. The stochastic simulation algorithm uses simulation (or Monte Carlo techniques), and gives approximate values [32,71,88]. The accuracy depends on the size of the sample space (the number of simulation runs). The evaluation suggested there does not seem to be a single algorithm, either exact or approximate, that works well for all kinds of networks. Each algorithm mentioned above has computational properties that render it attractive for probabilistic reasoning on certain kinds of network structures. As far as computational complexity is concerned, the number of loop cut nodes is a crucial factor for Pearl's algorithm. On the other hand, the size of the largest clique in the L-S algorithm is the key factor. In general, the size of the largest clique in the general graph should not be large, in particular, the graph should be large and sparse. The L-S algorithm is then expected to perform well. The stochastic simulation algorithm suffers slow convergence when there are extreme conditional probabilities. The problem was analysed and a possible solution proposed in section 2.4.4.2.

Based on study and evaluation of different reasoning algorithms in chapter 2, this thesis then shows how flexible probabilistic reasoning and learning system may be constructed and applied to uncertainty management in knowledge based systems. In particular, a probabilistic reasoning expert system shell,

PRESS [33], was designed and implemented (chapter 3). PRESS is primarily a research tool and implements many of the latest developments on Bayesian belief networks. It combines probabilistic modelling techniques with a front end that offers the flexibility and expressive power of a graphical environment. User interface issues are concerned with how to provide a "friendly" human-computer interface so that it is easy for people to operate or experiment with the system. The system is highly interactive and allows the design of the models and performance of uncertain reasoning via a simple graphical interface.

In comparison with other developed systems, for example HUGIN [56], PRESS offers control facilities, which give the measure of a node of particular interest and helps the user to control and understand the probabilistic reasoning process. It shows how balanced the actual advice given by the system is and the influence of evidence as it has been flowing around the network. A measure is provided to support decisions to direct user requests in order to optimise a specific goal satisfaction. Furthermore, PRESS provides a choice of algorithms for probabilistic reasoning, including exact and approximate methods.

The development of PRESS provides a framework to experiment with exact and approximate algorithms discussed in chapter 2. The experiments showed the differences between Pearl's and the L-S algorithms are significant. For example, in one of the experiments Pearl's algorithm takes 49 seconds to initialise a network but the L-S algorithm takes only 0.87 second to do so. The experiments also confirmed that exact computations suffer an exponential computational complexity [80]. In contrast with exact algorithms, the computation of the stochastic simulation algorithm has polynomial complexity in the size and number of connections in the graph.

The open architecture of PRESS made it possible to incorporate the extension of the algorithms and communicate with an external environment. The system was extented to handle a mixture of continuous and discrete variables in chapter 4 [33,34]. The differences between the computational procedure for pure models and that for mixed models were addressed. The approximation of the marginal distribution of a clique by a conditional Gaussian distribution whose moment characteristics agree with the true marginal moments was carried out [34]. While this was an extension of the application of Bayesian belief networks there are still a number of assumptions which limit the immediate value of the algorithm in some

applications, for example, in the forecasting context. As well as the restrictions to conditional Gaussian distribution, the method requires that no continuous variable has discrete 'children' in the network, and also, that the interactions between continuous nodes and other nodes can be described in terms of linear functions [34].

As a result of these developments, PRESS presents a systematic way of handling probabilistic reasoning using Bayesian belief networks. The system deals with a range of problems, including both pure and mixed probabilistic models. Examples from forensic science [97,104] and the forecast of a crop yield [35] were used for evaluation of the reasoning procedures in chapter 3 and chapter 4. Although these examples are not large enough, they illustrate the basic concepts of probabilistic reasoning using Bayesian belief networks and the main features of PRESS.

In a Bayesian belief network setting, expert systems could connect to existing databases and improve by gaining experience through parameter learning [36]. It is also possible to discover a Bayesian belief network in a semi-automated process by extracting the underlying topology directly out of the data, that is, using structure learning techniques [29,130] to avoid the bottleneck problem [131]. These two issue were addressed in chapter 5. In particular, sequential learning technique on Dirichlet distributions [36] and polytree algorithm [29] were investigated. Applications on a large medical database (6387 patient records) have been made. Two computational systems were set up for classification using PRESS, where one of them included the sequential learning technique. The comparison between the two systems showed that the performance of the system with learning was much better than that of the system without learning in the sense of overall accuracy when the training set is small (<500). When the size of the training set is large, the performance of the two systems are at the same level. The structure constructed by the polytree algorithm from the database was used for classification using PRESS. The overall accuracy (68%) of the classification is comparable with other methods (e.g. G&T, ID3) [22,38]. Furthermore, these experiments enabled us to identify problems associated with learning in Bayesian belief networks and future research directions.

6.3. Future Research

An attempt to study, develop and apply probabilistic reasoning and learning using Bayesian belief networks is made in this thesis. However, some further research in these areas is required. It is believed that progress in any of these areas may lead to a more successful development and a wider use of PRESS in solving real problems.

6.3.1. Modelling Dynamic Worlds

Modelling dynamic systems has not been addressed in this thesis. The idea of representing a dynamic world by using a Bayesian belief network (BBN) formalism would be interesting, so that pieces of BBN representing the world can be embedded in larger BBN representing more generic probabilistic relationships between the variables in the problem.

Time plays an essential role in a dynamic system. The representation of temporal knowledge and temporal reasoning will be crucial aspects in modelling dynamic systems [107,108]. One way of representing processes that unfold over time is by defining a finite state space and by modelling progression as a series of transitions between these states [132]. In such a way the system is able to support various types of temporal inference, such as predicting the future consequences of an action, or reconstructing past evolution based on incomplete information about the past and about the present, or detecting deviations from expected progress. The computational difficulties associated with propagation algorithms necessary to perform the inference have to be tackled.

6.3.2. Semi-Automatic Construction Models From Database

One alternative way of constructing an initial model is to exploit a database of previous cases from the domain. Recent research progress creates an opportunity now to discover knowledge in the form of general graphical "structure" of a problem, or underlying topology in Bayesian belief networks [133,134].

A theory or methodology for data-driven discovery of causation needs to be developed. Interactive graphical tools for constructing and manipulating the model would be essential. Feasible procedures for

determining all of the models statistically equivalent to any given model without latent variables needs to be developed. The genuine causal influences should be distinguished from spurious covariations [133,134]. The statistical and temporal aspects of causation should be addressed fully [135,136].

Furthermore, we do not possess a reliable informative procedure for constructing initial latent variable models from data alone, nor do we possess a solution to the closely connected problem of feasibly generating all latent variable causal models statistically equivalent to a given model. These problems would require further fundamental research.

6.3.3. Dynamic Improvement of Models

Dynamic improvement of the qualitative structure of the model as a database of cases accumulates will be a vital area for future research. It would involve criticisms of the model and possibly modifying it. We might also incorporate devices for monitoring the adequacy, in terms of better predictions, of a given model and/or of the prior distributions embodied in it to the problem at hand.

Data-based choice among graphical models has been studied while the existence of proper priors supports the use of Bayes factors to compare models of differing dimensionality. In expert systems a global comparison for two competing structures S_1 and S_2 could be based on the Bayes factor [128,129] or the minimum message length [137]. It should be possible to decompose the global comparison into components relative to, for example, cliques of the triangulated graph. Hence local model comparisons, concerned with whether a link could be dropped, or whether two adjacent cliques should be merged, could be monitored locally. It could allow local model criticism.

An important point is whether the assumptions exploited in the initial graphical models are compatible with a set of observed data. A possible solution is to provide a number of possible graphical models, and compare them in terms of the support they are given by data. Differences between various models can be interpreted in terms of the coding approach of Rissanen [138]: the preferred model amongst a class of competing models will be the one with the shortest total encoded length minimised with respect to its parameter estimates.

6.3.4. Parallel Implementation

Those reasoning algorithms discussed in chapter 2 have inherent parallelism and implementations could be in parallel processing environments. For example, the computational structure of a clique tree in the L-S algorithm lends itself to local message passing between autonomous processors [28]. Parallel processing would also be suitable for the stochastic simulation algorithm. Each node in the network could be assigned to a separate processor to carry out local computations and simulations.

Appendix A

BASIC GRAPH THEORY

In this appendix, basic notations in graph theory are reviewed. Our emphases will be on triangulated graphs and how to make a graph triangulated. The notation of triangulated graphs applies to problem solving within as widely different areas as solution of sparse symmetric systems of linear equations [139], pedigree analysis [140,141], and evidence propagation in Bayesian belief networks [25,28,33,56]. Our interest in graph triangulation originates from the latter of these areas. Most of the concepts reviewed have been studied extensively in the graph theory literature (see [142,143]).

Notations and Terminology

A graph, denoted G, is a pair (V,E), where V is a finite set of nodes and E is a set of edges between pairs of nodes. We will use n=|V| to denote the number of nodes and e=|E| to denote the number of edges in the graph. In general, the edges could be either directed or undirected. If all edges of a graph are directed, we call it a directed graph. If the directions of the arrows in a graph are not specified, such a pair (V,E) is called an undirected graph. The graph in which we are interested will not have both types of edges. We will represent a graph with nodes as circles and edges as either arrows or lines, see Figure A.1 and Figure A.2 as an example of directed and undirected graphs respectively. Let S be a set of nodes. Then G(S)=(S,E(S)) is the subgraph of G induced by S, where $E(S)=\{\{v,w\}\in E \text{ and } v,w\in S\}$.

If there is $u\rightarrow w$, u is a parent of w and w is a child of u. We will use PA(w) to represent all the parent nodes of w. Two nodes $u, w\in V$ are adjacent (or neighbours) if $\{u,w\}\in E$. We will use Adj(u) to represent all nodes adjacent to u in the graph G. In a directed graph, the neighbours of a node are its parents and children. For example, the directed graph depicted in Figure A.1 shows PA(Y)={D,X,H} and Adj(D)={B, Y}. A path between w and u is a sequence of nodes $(v_1, v_2, ..., v_k)$ with v_1=w and v_k=u so that each node in the sequence has one endpoint in common with its predecessor and its other endpoint in common with its

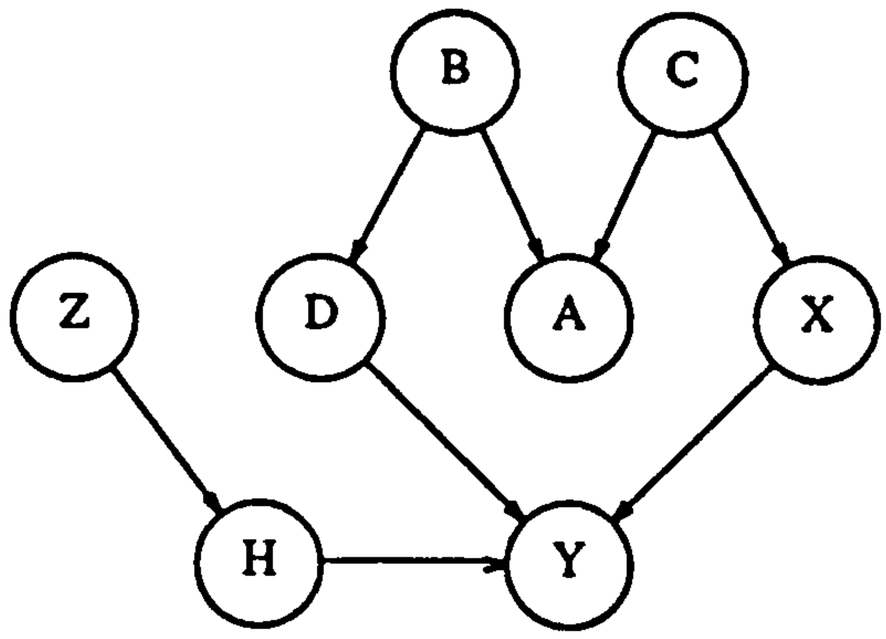


Figure A.1. Directed Graph

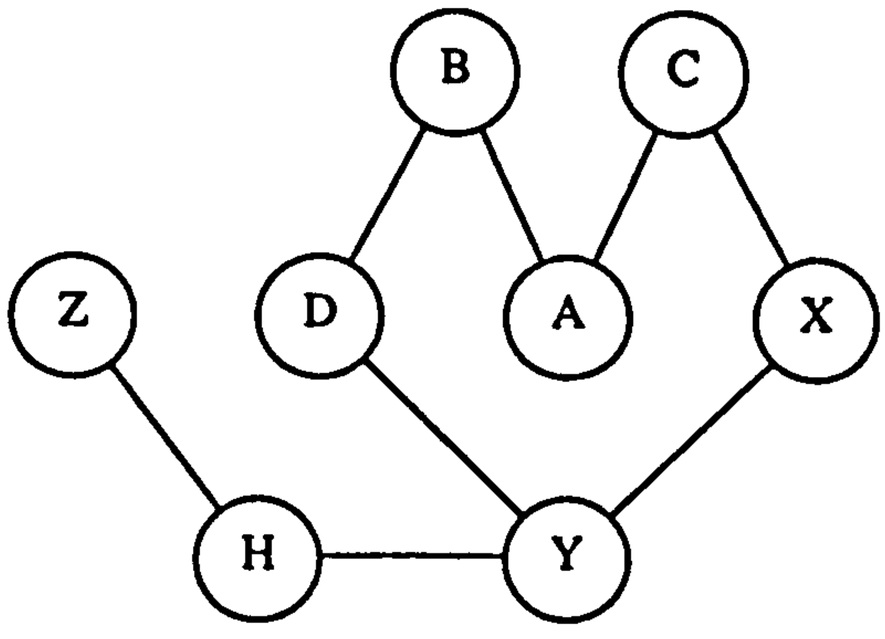


Figure A.2. Undirected Graph

successor. The number of nodes in the sequence is the length of the path. A loop (or a cycle) is a path (v_1, v_2, ..., v_k) where (i) no edge appears twice in the sequence (ii) the two endpoints are the same nodes, i.e. $v_1=v_k$. For example, there is an undirected loop (A, B, D, Y, X, C, A) in Figure A.2. Throughout this appendix, we assume that the graphs considered are connected, that is, for each pair of distinct nodes u, w, there is a path with these nodes as its endpoints. A directed acyclic graph (DAG) is a directed graph without any *directed* loops, e.g. Figure A.1. The directed acyclic graph plays an important role in probabilistic reasoning using graphical knowledge representations. A moral graph is obtained by joining parents of a directed graph, i.e. by adding edges u–w between all pairs u,w∈PA(x) where neither u→w nor w→u, and then dropping directions of the graph. The moral graph G_M for Figure A.1 is shown in Figure A.3, where dotted lines are additional edges. A tree is defined to be a connected graph without a loop. A path connecting two nodes is always uniquely determined in a tree. A polytree, known as a singly connected structure, is a directed structure which does not have any loops (as in Figure A.4) [29].

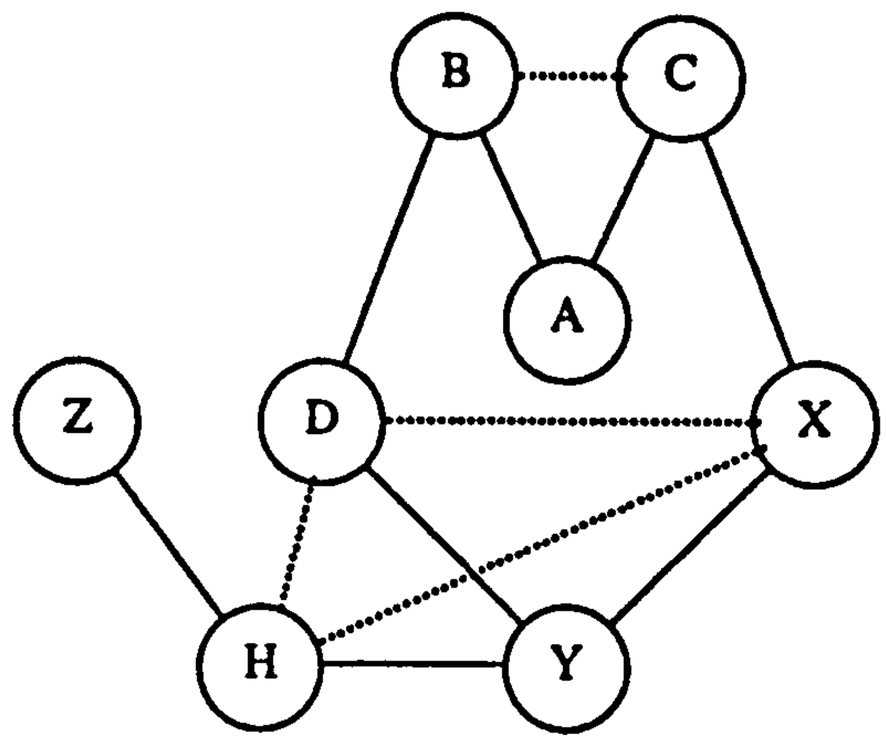


Figure A.3. Moral Graph

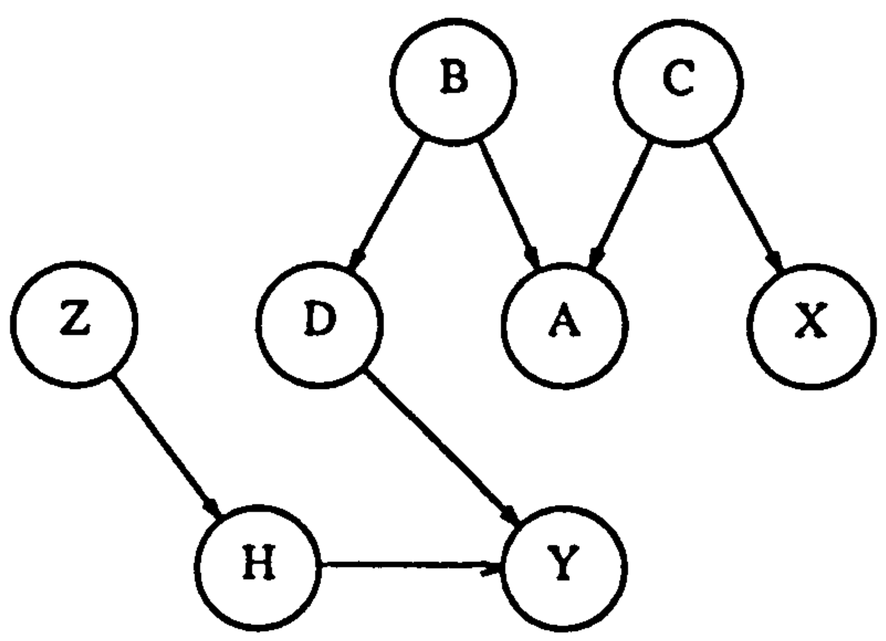


Figure A.4. Polytree Structure

Triangulated Graph

A triangulated graph is an undirected graph which has the property that all loops ($u=v_0, v_1, ..., v_k=u$) of length k>3 possess a shortcut (i.e. an edge joining two non-consecutive nodes of the loops) [142,143]. An example of triangulated graph is given in Figure A.5.

Triangulated graphs are also called rigid circuit graphs [144], chordal graphs [145] and decomposable graphs [74]. A subgraph of a triangulated graph G_T is also a triangulated graph because, otherwise, it would have a loop of length k>3 without a shortcut, and G_T would also have a loop of length k>3 without a shortcut. A tree is a triangulated graph. A subset of the undirected graph is complete if all the nodes of the subset are pairwise connected. A clique is defined as a maximal subset which is complete in an undirected graph. For example, there are 5 cliques in Figure A.5. They are {A,B,C}, {B,C,X}, {B,D,X}, {D,X,Y,H} and {Z,H}.

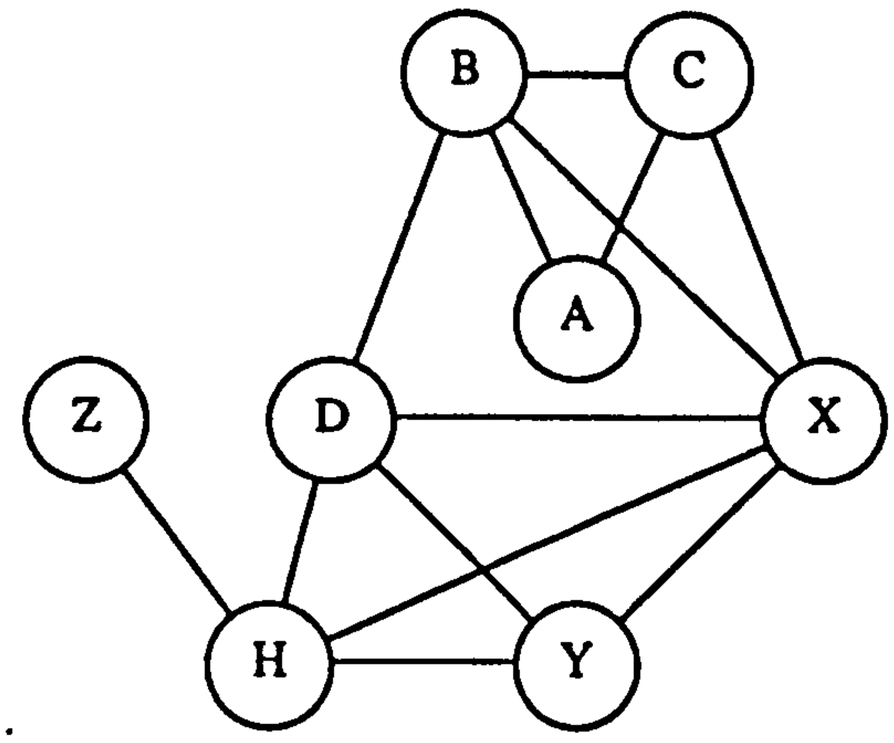


Figure A.5. Triangulated Graph

An efficient method for checking whether a graph is triangulated or not is based on ordering nodes in a special way. A ordering of V is a bijection ω: V$\leftrightarrow${1,2,...,n}. Let G=(V,E) be an undirected graph having n nodes. An ordering $\omega=\{v_1, v_2, ..., v_n\}$ of all nodes in V is called *perfect* if for every v_i, $Adj(v_i)\cap\{v_1,v_2,...,v_{i-1}\}$ is a complete subset of G. A possible perfect ordering for Figure A.3 is shown in Figure A.6. An undirected graph is triangulated if and only if it admits a perfect ordering. If an undirected graph G in which V has a perfect ordering $\omega=\{v_1, v_2, ..., v_n\}$ and there are *m* cliques, and if $\{C_1, C_2, ..., C_m\}$ be an ordering of the cliques according to their highest labelled nodes, then the ordering of the cliques has the *running intersection property* [28], that is, for every clique C_j (j>1), there exists an i<j such that clique C_i will have

$$C_j \cap (C_1 \cup C_2 \cup \cdots \cup C_{j-1}) \subseteq C_i .$$

C_i is known as a parent clique of C_j. For the graph shown in Figure A.6, we will have the following table to illustrate the running intersection property.

Order	Clique	Highest Label	Intersection	Parent Clique
1	{A, B, C}	3		
2	{B, C, X}	4	B, C	1
3	{B, D, X}	5	B, X	2
4	{D, X, Y, H}	7	D, X	3
5	{Z, H}	8	H	4

The basic technique applied to make a graph G triangulated is to add the extra edges F provided by eliminating the nodes of G one by one. Therefore the triangulated graph corresponding to G is represented by G_T=(V, E$\cup$F). A node V_i is eliminated by adding edges such that the nodes adjacent to V_i are pairwise

connected and by subsequent deletion of V_i and its incident edges. Triangulating graphs using the eliminating technique is essentially a problem of establishing a sequential order of the nodes specifying the ordering in which they should be eliminated.

The two most well-known ordering algorithms are those of maximum cardinality search by Tarjan and Yannakakis [146] and of lexicographic search by Ross, et al [147]. Maximum cardinality search is a method for ordering the nodes of an undirected graph, which always give a perfect ordering if the graph is triangulated. The lexicographic search does something similar, but in addition it is guaranteed to give minimal triangulation in O(ne) time, where as the maximum cardinality search is not. The set F of fill edges is minimal if for any $F' \subset F$, $G'=(V, E \cup F')$ is not triangulated. F is minimum if for any triangulation F' of G, $|F| \leq |F'|$. However, minimal orderings are not necessarily close to a minimum fill ordering. Finding a minimum fill ordering for an arbitrary graph is NP-complete† problem [148]. A stochastic algorithm based on simulated annealing by which the optimal solution to finding a minimum fill ordering problem may be found is presented and discussed in [78].

Maximum cardinality search has the property of not adding edges to an already triangulated graph and hence it may be used as an efficient test for graph triangulation as it runs in O(n+e) time. If an undirected graph is not triangulated, it can be made so by adding edges between nodes. Let G=(V,E) be an undirected graph and ω be a total ordering of the nodes in V, $F(\omega)$ is a set of (u,w) such that $(u,w) \notin E$ and there is a path between u and w containing only u, w and nodes ordered after both u and w. That is, if x is a node on the path other than u and w, then $\omega(x)>\omega(u)$ and $\omega(x)>\omega(w)$. $F(\omega)$ is called the fill-in of G with respect to ω. An algorithm for computing fill ins in a given undirected graph can be described as follows:

(1) Compute an ordering for the nodes. For example, using a maximum cardinality search, i.e., number nodes from 1 to n, in increasing order, always assigning the next number to the node having the largest set of previously numbered neighbours (breaking ties arbitrarily).

(2) From i=n to i=1, recursively fill in edges between any two nonadjacent parents of V_i, i.e., neighbours of V_i having lower ranks than V_i (including neighbours linked to V_i in previous steps). If no edge is

---

†NP is the class of problems that can be solved nondeterministically in polynomial time on a Turing machine. The "hardest" problems in NP are the NP-complete problems. No NP-complete problem is going to yield to a polynomial time algorithm. Moreover, if any NP-complete problem did yield to such an algorithm then so would every problem in NP.

added the original graph is triangulated; otherwise, the new filled graph is triangulated.

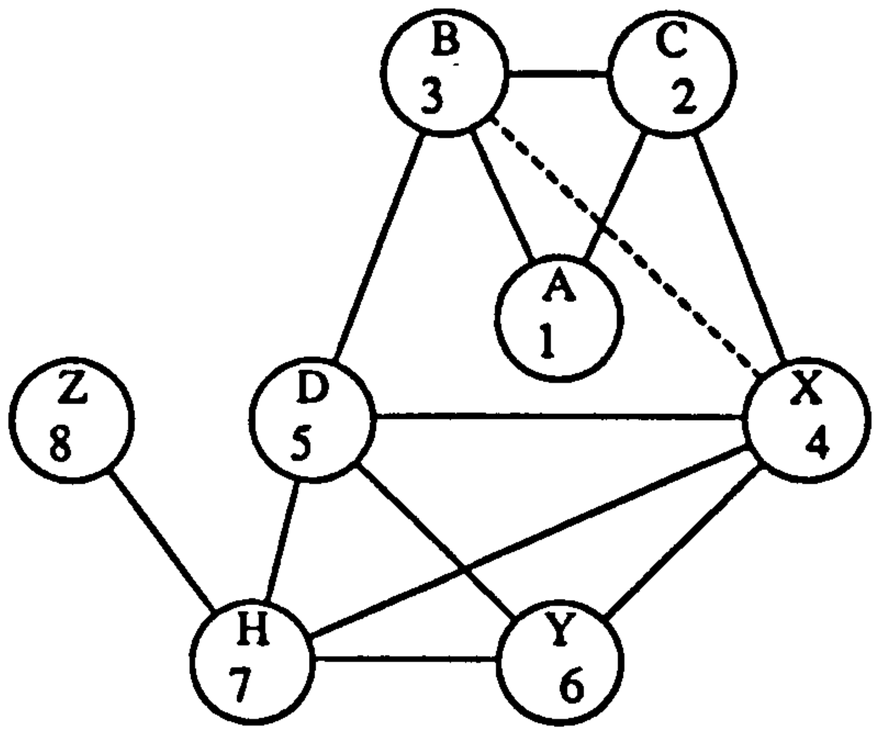


Figure A.6. A Triangulated Graph Derived from Figure A.3

The result of applying the maximum cardinality search to an undirected graph (see Figure A.3) is given in Figure A.6, where the dashed line represents the possible fill in.

If u and w are distinct nodes with $\{u,w\} \notin E$ in a graph, a set of nodes S is called a separator of u, w if every path between u and w contains a node in S. If U, W and S are pairwise disjoint sets of nodes, we say that S separates U and W if S is separator of u, w for all $u \in U$ and $w \in W$. Let A and B be a set of nodes in the graph G and $A, B \subseteq V$, with $A \cup B = V$. Then (A,B) is a decomposition of G if

(1) $S = A \cap B$ is complete;

(2) S separates A from B in G.

(A,B) is a proper decomposition if neither A nor B = V. A graph G=(V, E) is decomposable if either (1) V is complete; or (2) G has a proper decomposition (A,B) with G_A, G_B each decomposable. This is equivalent to the triangulation of the graph.

Darroch, Lauritzen and Speed [74] show how "decomposable" models (those whose joint probability distribution can be expressed wholly in terms of the marginal distributions on the cliques) correspond to triangulated graphs. We have a Markov random field defined on the moral graph, and we may use the independence properties derived from undirected graphs that are more usually exploited in applications such as image processing [149]. Triangulated graphs ensure that from now on it will be sufficient to derive the appropriate clique marginals. Essentially we have embedded our original structure in one that is more complex, in that some of the conditional independence assumptions are no longer apparent from the graph,

but allows much simpler analysis.

Conditional Independence and Bayesian Belief Networks

Conditional independence is based on a set of qualitative properties [150,151]. These properties allow us to deduce new conditional independences without explicit reference to the probability specifications [152,153,154]. For the sake of simplicity, we will use .. ⊨ .. | .. to mean ".. is conditionally independent of .. given .." defined on all three disjoint sets X, Y and Z, where X, Y, Z are subsets of a set of random variables. Let us look at the following axioms which are the basis for validating a new independence:

(1) Symmetry

$A \models B \mid C \Leftrightarrow B \models A \mid C$

(2) Decomposition

$A \models B \mid C$ and $D \subseteq A \Rightarrow D \models B \mid C$

(3) Weak Union

$A \models B \mid C$ and $D \subseteq A \Rightarrow A \models B \mid (C,D)$

(4) Contraction

$A \models B \mid C$ and $A \models D \mid (C,B) \Rightarrow A \models (B,D) \mid C$

These axioms form a system called *semi-graphoid* [155,156]. These axioms are very similar to those assembled by Dawid [150] for probabilistic conditional independence and those proposed by Smith [157] for generalised conditional independence.

An axiomatic basis is established so that we can infer new independencies by non-numerical, logical manipulation. Such axioms could serve as building blocks of expert systems that provide qualitative explanations as to why certain facts were or were not taken into account in a given computation. Currently, it appears unlikely that there exists a finite set of axioms which is complete for conditional independence.

The concept of separation in graphs has similar properties [155,156]. In addition, the graph has properties of explicitness, saliency and stability. The graphical representation provides a comfortable way to express conditional independencies explicitly [156]. Therefore it is a kind of language for expressing conditional independencies and a powerful tool for handling conditional independence in probabilistic reasoning. Moreover conditional independence provides a basis for expressing the qualitative structure in graphical form as a Bayesian belief network.

A Bayesian belief network can be viewed as an economical scheme for representing conditional independence relationships and for deducing new independencies from the network. It is of crucial importance to have effective and simple criteria for deciding which information or facts have dependencies upon others in the light of present knowledge and the reading of all conditional independence directly from the Bayesian belief network. The following definition permits us to graphically identify and validate such conditional independence from a Bayesian belief network. If X, Y, Z are three disjoint subsets of nodes in a Bayesian belief network, then Z is said to *d-separate* X from Y, if and only if there is no path from a node in X to a node in Y along which the following two conditions hold

(1) a node z in Z and arrows do not meet head to head at z;

(2) a node z not in Z, nor has z any children in Z, and arrows of path meet head to head at z.

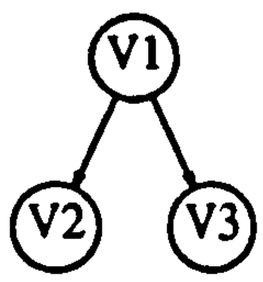
This is known as *d-separation* criterion, discovered by Pearl [154]. If X and Y are d-separated by Z, then $X \models Y \mid Z$. Lauritzen et al proposed another criterion (the *directed global Markov property*) for conditional independence [76]. They investigated conditional independence properties of directed Markov fields and showed the result using the trick of forming the moral graph of a Bayesian belief network with known facts about Markov fields over undirected graphs. The directed global Markov property states that $X \models Y \mid Z$ whenever X and Y are separated by Z in the moral graph of the smallest ancestral set containing $X \cup Y \cup Z$. It has been proved that this criterion is equivalent to that of Pearl [154].

Bayesian belief networks have a close relationship with conditional independence in probability theory. In a Bayesian belief network, if A, B and C are three disjoint subsets of variables of a probability distribution P, we have $A \models B \mid C \Longleftrightarrow P(X_A \mid X_B, X_C) = P(X_A \mid X_C)$ for all values X_A, X_B and X_C such that $P(X_A, X_C) > 0$ [150]. The marginal independence can be expressed as $A \models B \mid \varnothing \Longleftrightarrow P(X_A \mid X_B) = P(X_A)$

for all X_A and X_B such that $P(X_B)>0$.

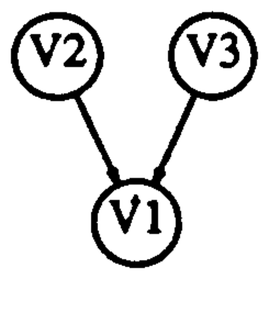
There are basic Bayesian belief networks that show different kinds of conditional independence. Without loss of generality, we consider three variables V_1, V_2 and V_3. There are four possible different cases.

Case 1: V_1 is a common cause of V_2 and V_3. The directed graph is:



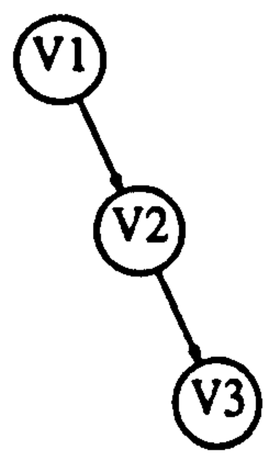
The dependence read from the graph is: $V_2 \models V_3 | V_1$. In other words, $P(V_2, V_3 | V_1)=P(V_2 | V_1)P(V_3 | V_1)$.

Case 2: V_1 is a common effect of V_2 and V_3. Their relations are depicted as:



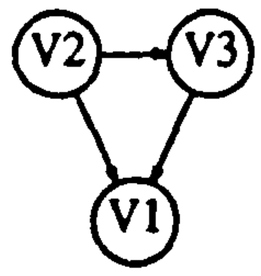
We have $V_2 \models V_3 | \varnothing$. Therefore $P(V_2, V_3)=P(V_2)P(V_3)$.

Case 3: V_1 causes V_2, and V_3 is an effect of V_2. The graph is:



The graph gives information: $V_1 \models V_3 | V_2$. The joint probability distribution can be expressed as: $P(V_1, V_2, V_3)=P(V_3 | V_2)\, P(V_2 | V_1)P(V_1)$.

Case 4: V_2 causes V_1 and V_3. V_1 is an effect of V_3. The graph is:



From the graph, no conditional independence can be read. Therefore, we have the joint distribution $P(V_1, V_2, V_3)=P(V_2)P(V_1|V_2, V_3)P(V_3|V_2)$.

Appendix B

CONVERGENCE PROBLEM IN STOCHASTIC SIMULATIONS

Convergence problems have been discussed in the simulation literature [158,159] and they remain an active area of research. In particular, Wilson [158,160] separates the convergence problem into two components, initialisation and steady state behaviour. The former concerns initialising the states of variables in a model to promote rapid convergence to the posterior probabilities. The latter deals with how to reach convergence as quickly as possible, given an initial state. In this appendix, we focus primarily on the steady state problem. Note that the performance of Pearl's stochastic simulation method may be quite sensitive to initialisation of the network as well [89]. The main problem in applying the stochastic simulation method to Bayesian belief networks is that the method provides no variance or error estimates on posterior probabilities. Thus it may be difficult to know how long to simulate in order to achieve acceptably accurate estimates of the posterior probabilities of interest. It does not offer prior upper bounds on the amount of computation that will guarantee sufficient convergence properties. The theory of random Markov fields guarantees convergence to the correct stationary distribution in the limit. There is no reason, however, to believe that stochastic simulation method converges rapidly or efficiently to that distribution. To demonstrate, the example given in section 2.4.4.2 will be employed. A possible solution to this problem is proposed.

It can be seen from Figure 2.7 in section 2.4.4.2 that the whole sample space is divided into two isolated chains, chain 1 and chain 2. This is because there are conditional probabilities that are 1 or 0 in the example. In chain 1, there is only one state (¬s,¬t,¬e). That is,

$$P_1(s)=0, \quad P_1(t)=0, \quad P_1(e)=0.$$

Chain 2 consists of three states: (s,t,e), (s,¬t,e) and (¬s,t,e). For convenience, we will call them state 1, state 2 and state 3, respectively.

Suppose an infinitely long simulation process is conducted. The simulation process has the Markov property, that is, the new states of nodes are dependent only on the preceding states. It is, therefore, easy to determine the probability of a node being in a given state during an infinite simulation.

Assume that we are now in state 1, that is, (s,t,e) and we process nodes in the order of T, S and E. We have:

$$P(t^* \mid X_T) = \alpha P(t) P(e \mid s,t) = \alpha\, 0.4 = 2/5,$$

$$P(\neg t^* \mid X_T) = \alpha P(\neg t) P(e \mid s,\neg t) = \alpha\, 0.6 = 3/5,$$

where α=1. So there is a probability of 2/5 for T to be *true* and of 3/5 for it to be *false*. Let us assume that T is chosen as *true*. We process S:

$$P(s^* \mid X_S) = \alpha P(s) P(e \mid s,t) = \alpha 0.2 = 1/5,$$

$$P(\neg s^* \mid X_S) = \alpha P(\neg s) P(e \mid \neg s,t) = \alpha 0.8 = 4/5,$$

where α is 1. In this case S is unlikely to be *true*. If we assume that T takes *false* instead of *true*, it will yield

$$P(s^* \mid X_S) = \alpha P(s) P(e \mid s,\neg t) = 1,$$

$$P(\neg s^* \mid X_S) = \alpha P(\neg s) P(e \mid \neg s,\neg t) = 0.$$

where α=5. This means that S will definitely take *true*. When the values of S and T are given, the value of E will definitely be known from the conditional probability tables.

To sum up, we can possibly go to state 2, 3 or stay in state 1 from state 1 with the following probabilities

state 1 → *state* 1: 2/5×1/5=2/25,

state 1 → *state* 2: 3/5×1=3/5,

state 1 → *state* 3: 2/5×4/5=8/25.

Similarly we can work out the possible state transformations and their probabilities from state 2 and state 3. The conclusions are

state 2 → *state* 1: 2/25, and *state* 3 → *state* 1: 1/5,

state 2 → *state* 2: 3/5, *state* 3 → *state* 3: 4/5.

state 2 → *state* 3: 8/25,

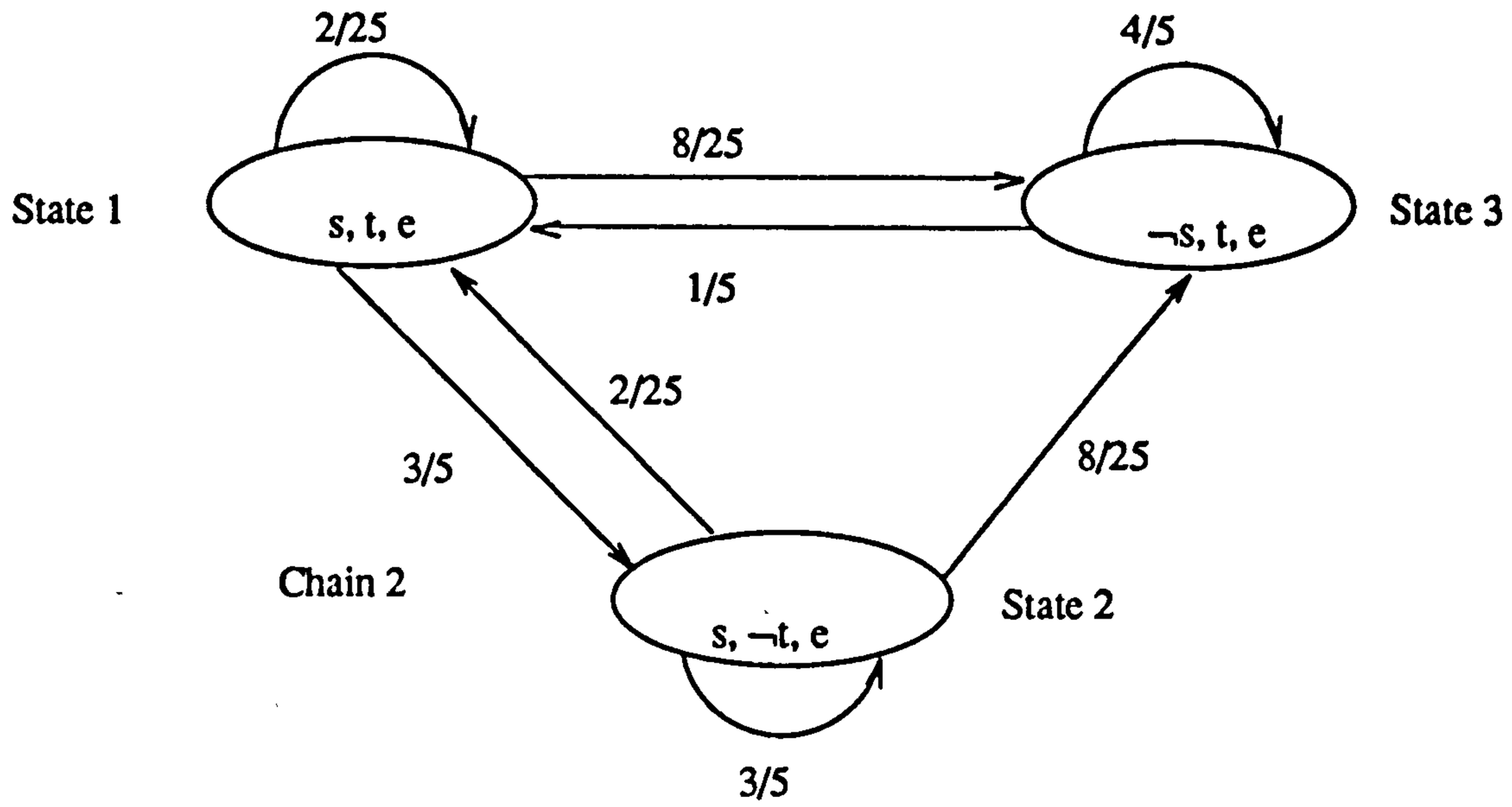


Figure B.1. State Transformations with Their Probabilities in Chain 2

All these state transformations and their probabilities are shown in Figure B.1. It is possible that we can calculate the probability of a proposition being in a particular state from Figure B.1. For example, state 1 can be transferred from state 2, state 3 and its own. In fact, we have

P(*state* 1)=2/25P(*state* 1)+2/25P(*state* 2)+1/5P(*state* 3),

P(*state* 2)=3/5P(*state* 1)+3/5P(*state* 2),

P(*state* 3)=8/25P(*state* 1)+8/25P(*state* 2)+4/5P(*state* 3).

So numerical relations among these three states are P(*state* 2)=3/2P(*state* 1) and P(*state* 3)=4P(*state* 1). Using the probability property

P(*state* 1)+P(*state* 2)+P(*state* 3)=1,

we obtain

P(*state* 1)=2/13 that is: P(s,t,e)=2/13

P(*state* 2)=3/13 P(s,¬t,e)=3/13

P(*state* 3)=8/13 P(¬s,t,e)=8/13.

The probability of each proposition being *true* can be derived by

$$P_2(s) = P(s,t,e) + P(s,\neg t,e) = 5/13,$$

$$P_2(t) = P(s,t,e) + P(\neg s,t,e) = 10/13,$$

$$P_2(e) = P(s,t,e) + P(s,\neg t,e) + P(\neg s,t,e) = 1.$$

The states can be transformed inside the chain, but cannot lead to other chain. If we just apply stochastic simulation approach directly, we consider only one of the possible chains. In our example, we consider either chain 1 or chain 2. Therefore, it is impossible to obtain reasonable estimated probabilities in this case because only a part of the sample space has been taken into account. The generated sample space will not be complete. It is impossible that the sample space generated by the stochastic simulation algorithm will cover all the four possible states if it starts from an initial state.

We have to generate a sample space covering all possible states in order to converge reasonable results. One way to deal with a model having conditional probabilities taking values 1 or 0 is to slightly change those conditional probabilities being 1 and 0. For example, probabilities set to 0 or 1 are changed into probabilities of, for example, 0.01 or 0.99 so that the gate between two set of states can be opened slightly. But the solution is not efficient if we do care about running time because, if the gate is open, a lot of iterations are needed to go through it. It is not clear what effect this ad hoc adjustment will have on the overall performance of a system.

Another possible solution is to apply the stochastic simulation algorithm to all possible isolated chains. Then their simulated results are weighted by the probabilities with which chains occur in order to construct a complete sample space. In other words, a small sample space is generated for each isolated chain. These small sample spaces are mutually exclusive because these chains cannot communicate with each other. Then a global sample space is built from these small sample spaces proportionally. Therefore, a sample space covering all possible states is constructed in this way. Based on the idea above, Pearl's algorithm can be slightly modified and a possible solution looks like:

Analyse the causal model
FOR each isolated chain C_i DO
  Apply Pearl's algorithm to chain C_i
  Calculate probability P(C_i) from given conditional probability tables
END
Sum all the results weighted by the probability P(C_i)
Calculate overall estimated probabilities

Considering our example, the probabilities of getting from chain to chain can be calculated from Table 2.1, that is, we start from chain 1 with probability 48% (i.e. P(¬s)P(¬t)). and there is a probability of 52% of starting from chain 2. In other words, we have P(*chain* 1)=0.48 and P(*chain* 2)=0.52. Then sum the two results P_1 and P_2 weighted by the probability P(*chain* 1) and P(*chain* 2) respectively, that is, P(S)=P_1(S)P(*chain* 1)+P_2(S)P(*chain* 2). Therefore the overall probabilities are:

$P(s)=5/13\times0.52+0\times0.48=0.20$,

$P(t)=10/13\times0.52+0\times0.48=0.40$,

$P(e)=1\times0.52+0\times0.48=0.52$.

These correspond to the correct marginal probabilities.

Appendix C

CG DISTRIBUTIONS AND CG POTENTIALS

This appendix describes the multivariate normal distribution and the conditional Gaussian (CG) distribution. We want to show operations on CG distributions and CG potentials in detail. Proofs of these operations are also given.

Multivariate Normal Distribution

The univariate normal density function of x can be written as $N(\mu,\nu)$ with mean μ and variance ν,

$$f(x) = (2\pi\nu)^{-\frac{1}{2}} exp\left(\frac{-(x-\mu)^2}{2\nu}\right), \tag{C.1}$$

where ν is positive. The density of function of a multivariate normal distribution of $x_1, ..., x_n$ has an analogous form (C.1). The scalar variable x is replaced by a vector

$$\mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},$$

the scalar constant μ is replaced by a vector

$$\xi = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix},$$

and the positive constant ν is replaced by a positive definite symmetric matrix

$$\Sigma = \begin{bmatrix} \nu_{11} & \nu_{12} & \cdots & \nu_{1n} \\ \nu_{21} & \nu_{22} & \cdots & \nu_{2n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \nu_{n1} & \nu_{n2} & \cdots & \nu_{nn} \end{bmatrix}.$$

Thus the density function of an n-variate normal (Gaussian) distribution is

$$f(x_1, ..., x_n) = ((2\pi)^n \ (det\ \Sigma))^{-\frac{1}{2}} exp(-\frac{1}{2}(X-\xi)^T \ \Sigma^{-1} \ (X-\xi)), \qquad (C.2)$$

where ξ is the mean vector and Σ is the covariance matrix of X. We shall denote the above distribution as $N(\xi, \Sigma)$.

Multivariate normal distributions have some useful properties. For example, marginal distributions and conditional distributions derived from multivariate normal distributions are also normal distributions. Moreover, linear combinations of multivariate normal variates are again normally distributed. Here we state some theorems without any proof [161].

Theorem 1:

If X is distributed according to $N(\xi, \Sigma)$, the marginal distribution of any set of components of X is a multivariate normal distribution with mean and covariance obtained by taking the corresponding components of ξ and Σ, respectively.

Theorem 2:

Let the components of X be divided into two groups subvectors X_1 and X_2. Similarly, the mean ξ is divided into ξ_1 and ξ_2, and the covariance matrix Σ of X is divided into Σ_{11}, Σ_{12}, Σ_{21} and Σ_{22}. Then if X is $N(\xi, \Sigma)$, the conditional distribution of X_1 given $X_2=x_2$ is normal with mean $\xi_1+\Sigma_{12}\Sigma_{12}^{-1}(x_2-\xi_2)$ and covariance matrix $\Sigma_{11} - \Sigma_{12} \ \Sigma_{12}^{-1}\Sigma_{21}$.

Theorem 3:

If X is distributed according to $N(\xi, \Sigma)$, then $Y=AX+C$ is a normal distribution $N(A\xi+C, A\Sigma A^T)$.

CG-Distribution

Assume that a finite set of variables V can be partitioned into discrete Δ and continuous Γ as $V=\Delta \cup \Gamma$. The random variables x take values in the product space consisting of discrete i and continuous y,

$$x = (x_\alpha)_{\alpha \in V} = (i, y) = \left\{ (i_\delta)_{\delta \in \Delta}, (y_\gamma)_{\gamma \in \Gamma} \right\},$$

where i_δ are finite sets of possible states of the discrete variables and y_γ are real valued. The corresponding random variables shall be denoted X_α, $\alpha \in \Delta \cup \Gamma$. Conditional Gaussian (CG) distributions [109,112] are multivariate distributions characterised by the joint conditional distribution of the continuous variables y, given a combination of discrete variables i as being Gaussian (Normal). Consider the following strictly positive density f

$$f(x) = f(i, y) = \chi(i) \exp\left(g(i) + h(i)^T y - \frac{1}{2} y^T K(i) y \right),$$

where i and y are vectors, $\chi(i) \in \{0,1\}$ indicates whether f is positive at i, $g(i)$ is a real-valued function of i, $h(i)$ is a vector-valued function of i, $K(i)$ is a matrix-valued function of i taking values in the set of positive definite symmetric matrices and v^T denotes the transpose of the vector v. Equivalently, the logarithm of the density f may be written as

$$\log f(x) = g(i) + h(i)^T y - \frac{1}{2} y^T K(i) y .$$

A probability distribution with density f defined above has a Conditional Gaussian distribution F in the sense that continuous variables y, for a given instance of discrete variables i, is multivariate Gaussian with covariance $K(i)^{-1}$ and expectation $K(i)^{-1} h(i)$, that is,

$$F(X_\Gamma \mid X_\Delta = i) = N_{|\Gamma|}(\xi(i), \Sigma(i)) .$$

Note that both the covariance and the expectation may depend on discrete variables i. The marginal distribution of the discrete variables X_Δ has probabilities $p(i)$,

$$p(i) = P\left\{ X_\Delta = i \right\} = (2\pi)^{-\frac{|\Gamma|}{2}} (det K(i))^{-\frac{1}{2}} \exp\left(g(i) + \frac{1}{2} h(i)^T K(i)^{-1} h(i) \right),$$

where det denotes determinant. Note that when $\Gamma = \varnothing$ there are only discrete variables. The only restriction then is that the probability should be positive. Similarly, when $\Delta = \varnothing$ all variables are continuous and the distribution is assumed to be multivariate Gaussian. In both cases, the models are pure.

A CG-distribution can be specified either by the triple $(g(i), h(i), K(i))$ or $\{p(i), \xi(i), \Sigma(i)\}$ if $K(i)$ is positive definite, whichever might be convenient in the context considered. These two triples are termed as the canonical and the moment characteristics of the CG-distributions respectively. $g(i)$, $h(i)$ and $K(i)$

denote the discrete, linear and quadratic canonical characteristics, respectively. $p(i)$ is probability, $\xi(i)$ is mean and $\Sigma(i)$ is covariance matrix. For more details on CG-distributions refer to Lauritzen and Wermuth [109].

CG-distributions have some useful properties [109]. However, CG-distributions are not closed under marginalisation in general since the conditional distribution of X_Γ given X_Δ can be a finite mixture of Gaussian distributions [109]. This illustrates the sense in which the multivariate family of CG-distributions is more complex than the family of Gaussian distributions. The latter is closed under conditioning as well as marginalising [161]. The difference between pure and mixed probabilistic models is due to this fact.

Operations on CG Distributions and CG Potentials

Here we consider CG distributions. In the case where Y is a continuous variable we specify the conditional distributions to be of the type

$$F(Y \mid PA(Y)) = \mathrm{N}(\alpha(i) + \beta(i)^T z, \gamma(i)), \qquad \text{(C.3)}$$

where z is a value of Z which is a vector of continuous parent variables of Y, $\alpha(i)$ is a real number, $\beta(i)$ is a vector and $\gamma(i)$ is a positive real number. The conditional density (C.3) then corresponds to a CG potential $\phi(i, y, z)$ defined on the space $I \times Y \times Z$ ($I \in \Delta$, $Y, Z \in \Gamma$ and $PA(Y) = I \cup Z$). The canonical characteristics $(g(i), h(i), K(i))$ can be derived, where

$$g(i) = -\frac{\alpha(i)^2}{2\gamma(i)} - \frac{1}{2}\log(2\pi\gamma(i)), \quad h(i) = \frac{\alpha(i)}{\gamma(i)}\begin{bmatrix} 1 \\ -\beta(i) \end{bmatrix}, \quad K(i) = \frac{1}{\gamma(i)}\begin{bmatrix} 1 & -\beta(i)^T \\ -\beta(i) & \beta(i)\beta(i)^T \end{bmatrix}.$$

PROOF:

Consider the density function (C.3),

$$\mathrm{N}(\alpha(i)+\beta(i)^T z, \gamma(i)) = (2\pi\gamma(i))^{-\frac{1}{2}} \exp\left(\frac{-(y-\alpha(i)-\beta(i)^T z)^2}{2\gamma(i)}\right).$$

Let $w = y - \beta(i)^T z$, then we find w is a random variable and

$$N(\alpha(i)+\beta(i)^T z, \gamma(i))$$

$$= exp\left(-\frac{1}{2}\log(2\pi\gamma(i)) - \frac{(w-\alpha(i))^2}{2\gamma(i)}\right)$$

$$= exp\left(-\frac{1}{2}\log(2\pi\gamma(i)) - \frac{(w^2 - 2\alpha(i)w + \alpha(i)^2)}{2\gamma(i)}\right)$$

$$= exp\left(-\frac{1}{2}\log(2\pi\gamma(i)) - \frac{\alpha(i)^2}{2\gamma(i)} + \frac{\alpha(i)(y-\beta(i)^T z)}{\gamma(i)} - \frac{(y-\beta(i)^T z)^2}{2\gamma(i)}\right). \quad (C.4)$$

On the other hand, the corresponding CG potential representation $\phi(i, y, z)$ can be expressed as:

$$\phi(i, y, z) = exp\left(g(i) + h(i)^T x - \frac{1}{2}x^T K(i)x\right),$$

where x is

$$x = \begin{bmatrix} y \\ z \end{bmatrix},$$

$h(i)^T x$ is

$$h(i)^T x = \frac{\alpha(i)}{\gamma(i)} \begin{bmatrix} 1 \\ -\beta(i) \end{bmatrix}^T \begin{bmatrix} y \\ z \end{bmatrix} = \frac{\alpha(i)(y-\beta(i)^T z)}{\gamma(i)},$$

and $x^T K(i)x$ is

$$x^T K(i)x = \begin{bmatrix} y \\ z \end{bmatrix}^T \frac{1}{\gamma(i)} \begin{bmatrix} 1 & -\beta(i)^T \\ -\beta(i) & \beta(i)\beta(i)^T \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix}$$

$$= \frac{1}{\gamma(i)} (y - \beta(i)z^T - y\beta(i)^T + z^T\beta(i)\beta(i)^T) \begin{bmatrix} y \\ z \end{bmatrix}$$

$$= \frac{1}{\gamma(i)} (y^2 - \beta(i)z^T y - y\beta(i)^T z + z^T\beta(i)\beta(i)^T z) = \frac{(y-\beta(i)^T z)^2}{\gamma(i)}.$$

So, we have the following expression for the above CG potential if $g(i) = -\frac{\alpha(i)^2}{2\gamma(i)} - \frac{1}{2}log(2\pi\gamma(i))$,

$$\phi(i, y, z) = exp\left(-\frac{\alpha(i)^2}{2\gamma(i)} - \frac{1}{2}log(2\pi\gamma(i)) + \frac{\alpha(i)(y-\beta(i)^T z)}{\gamma(i)} - \frac{(y-\beta(i)^T z)^2}{2\gamma(i)}\right). \quad (C.5)$$

The expression (C.5) is the same as that of (C.4). The result follows.

Multiplication and Division

Suppose that there are two CG potentials in the canonical characteristics form, $\phi_1(i, y)$,

$$\phi_1(i,y) = exp\left(g_1(i) + h_1(i)^T y - \frac{1}{2} y^T K_1(i) y \right)$$

and $\phi_2(i,y)$,

$$\phi_2(i,y) = exp\left(g_2(i) + h_2(i)^T y - \frac{1}{2} y^T K_2(i) y \right)$$

defined on the same space $I \times Y$ ($I \in \Delta$ and $Y \in \Gamma$). The multiplication is calculated

$$\phi_1(i,y) \times \phi_2(i,y)$$

$$= exp\left(g_1(i) + h_1(i)^T y - \frac{1}{2} y^T K_1(i) y \right) \times exp\left(g_2(i) + h_2(i)^T y - \frac{1}{2} y^T K_2(i) y \right)$$

$$= exp\left(g_1(i) + g_2(i) + (h_1(i) + h_2(i))^T y - \frac{1}{2} y^T (K_1(i) + K_2(i)) y \right).$$

So we have

$$(g_1(i), h_1(i), K_1(i)) \times (g_2(i), h_2(i), K_2(i)) = (g_1(i) + g_2(i), h_1(i) + h_2(i), K_1(i) + K_2(i)).$$

If $\phi_1(i,y) \neq 0$ and $\phi_2(i,y) \neq 0$, the division is calculated

$$\phi_1(i,y) + \phi_2(i,y)$$

$$= exp\left(g_1(i) + h_1(i)^T y - \frac{1}{2} y^T K_1(i) y \right) + exp\left(g_2(i) + h_2(i)^T y - \frac{1}{2} y^T K_2(i) y \right)$$

$$= exp\left(g_1(i) - g_2(i) + (h_1(i) - h_2(i))^T y - \frac{1}{2} y^T (K_1(i) - K_2(i)) y \right).$$

We have

$$(g_1(i), h_1(i), K_1(i)) + (g_2(i), h_2(i), K_2(i)) = (g_1(i) - g_2(i), h_1(i) - h_2(i), K_1(i) - K_2(i)).$$

Marginalisation

Assume that we marginalise a CG potential $\phi(i, x_1, x_2)$ with $(g(i), h(i), K(i))$ defined on (m+n) dimensions over continuous variables $X = X_1 \cup X_2$ and discrete variables I. The set of random continuous variables is partitioned into two parts so that $|X_1| = m$ and $|X_2| = n$. X_1 corresponds to those continuous variables we are going to marginalise. So X, $h(i)$ and $K(i)$ can be represented as:

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad h(i) = \begin{bmatrix} h_1(i) \\ h_2(i) \end{bmatrix}, \quad K(i) = \begin{bmatrix} K_{11}(i) & K_{12}(i) \\ K_{21}(i) & K_{22}(i) \end{bmatrix}.$$

The marginalised CG potential $\bar{\phi}(i, x_2)$ has the canonical characteristics $(\bar{g}(i), \bar{h}(i), \bar{K}(i))$ given as

$$\bar{g}(i) = g(i) + \frac{1}{2}(m\ log(2\pi) - \log(\det K_{11}(i)) + h_1(i)^T K_{11}(i)^{-1} h_1(i)),$$

$$\bar{h}(i) = h_2(i) - K_{21}(i) K_{11}(i)^{-1} h_1(i),$$

$$\bar{K}(i) = K_{22}(i) - K_{21}(i) K_{11}(i)^{-1} K_{12}(i).$$

PROOF:

Our task is to calculate $\int \phi(i, x_1, x_2)\, dx_1$. Consider the CG potential $\phi(i, x_1, x_2)$,

$$\phi(i, x_1, x_2) = exp\left(g(i) + \begin{bmatrix} h_1(i) \\ h_2(i) \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} K_{11}(i) & K_{12}(i) \\ K_{21}(i) & K_{22}(i) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$

$$= exp\left(g(i) + h_1(i)^T x_1 + h_2(i)^T x_2 - \frac{1}{2}(x_1^T K_{11}(i) x_1 + x_2^T K_{21}(i) x_1\right.$$

$$\left. + x_1^T K_{12}(i) x_2 + x_2^T K_{22}(i) x_2)\right).$$

Let $v(i) = -K_{11}(i)^{-1} K_{12}(i) x_2 + K_{11}(i)^{-1} h_1(i)$. Using the property that $K(i)$ is a symmetric matrix, then we find by direct calculation that

$$\phi(i, x_1, x_2) = exp\left(-\frac{1}{2}(x_1 - v(i))^T K_{11}(i)(x_1 - v(i))\right) \tag{C.6}$$

$$\times exp\left(x_2^T\ (h_2(i) - K_{21}(i) K_{11}(i)^{-1} h_1(i))\right) \tag{C.7}$$

$$\times exp\left(-\frac{1}{2} x_2^T\ (K_{22}(i) - K_{21}(i) K_{11}(i)^{-1} K_{12}(i))\, x_2\right) \tag{C.8}$$

$$\times exp\left(g(i) + \frac{1}{2} h_1(i)^T K_{11}(i)^{-1} h_1(i)\right). \tag{C.9}$$

This is because the item $(x_1 - v(i))^T K_{11}(i)(x_1 - v(i))$ in the expression (C.6) is

$$(x_1 - v(i))^T K_{11}(i)(x_1 - v(i))$$

$$= (x_1^T K_{11}(i) + x_2^T K_{12}(i)^T (K_{11}(i)^{-1})^T K_{11}(i)$$

$$- h_1(i)^T (K_{11}(i)^{-1})^T K_{11}(i))\, (x_1 + K_{11}(i)^{-1} K_{12}(i) x_2 - K_{11}(i)^{-1} h_1(i))$$

$$= (x_1^T K_{11}(i) x_1 + x_1^T K_{12}(i) x_2 - x_1^T h_1(i) + x_2^T K_{12}(i)^T\ (K_{11}(i)^{-1})^T K_{11}(i) x_1$$

$$+ x_2^T K_{12}(i)^T (K_{11}(i)^{-1})^T K_{12}(i) x_2 - x_2^T K_{12}(i)^T (K_{11}(i)^{-1})^T h_1(i)$$

$$- h_1(i)^T (K_{11}(i)^{-1})^T K_{11}(i) x_1 - h_1(i)^T (K_{11}(i)^{-1})^T K_{12}(i) x_2 + h_1(i)^T (K_{11}(i)^{-1})^T h_1(i))$$

$$= (x_1{}^T K_{11}(i)x_1 + x_1{}^T K_{12}(i)x_2 - 2h_1(i)^T x_1 + x_2{}^T K_{21}(i)x_1 + x_2{}^T K_{21}(i)K_{11}(i)^{-1}K_{12}(i)x_2$$

$$- 2x_2{}^T K_{21}(i)K_{11}(i)^{-1}h_1(i) + h_1(i)^T K_{11}(i)^{-1}h_1(i)) .$$

Note that only the expression (C.6) has x_1. Using the property of the probability density function,

$$\int ((2\pi)^m (det\ \Sigma(i)))^{-\frac{1}{2}} exp(-\frac{1}{2}(x_1 - v(i))^T \Sigma(i)^{-1}(x_1 - v(i)))\, dx_1 = 1 ,$$

therefore, we have

$$\int exp(-\frac{1}{2}(x_1 - v(i))^T K_{11}(i)(x_1 - v(i)))\, dx_1 = (2\pi)^{\frac{m}{2}} (det\ K_{11}(i))^{-\frac{1}{2}}$$

$$= exp(\frac{1}{2} (m\ log(2\pi) - \log(\det K_{11}(i)))) .$$

The new CG potential $\bar{\phi}(i, x_2)$ is calculated as

$$\bar{\phi}(i, x_2) = \int \phi(i, x_1, x_2)\, dx_1$$

$$= exp(g(i) + \frac{1}{2}(m\ log(2\pi) - \log(\det K_{11}(i)) + h_1(i)^T K_{11}(i)^{-1}h_1(i))$$

$$+ (h_2(i) - K_{21}(i)K_{11}(i)^{-1}h_1(i))^T x_2$$

$$- \frac{1}{2}x_2{}^T (K_{22}(i) - K_{21}(i)K_{11}(i)^{-1}K_{12}(i))x_2) .$$

This can be written in the form

$$\bar{\phi}(i, x_2) = \exp(\bar{g}(i) + \bar{h}(i)^T x_2 - \frac{1}{2}x_2{}^T \bar{K}(i)x_2) ,$$

where

$$\bar{g}(i) = g(i) + \frac{1}{2} (m\ log(2\pi) - \log(\det K_{11}(i)) + h_1(i)^T K_{11}(i)^{-1}h_1(i)) ,$$

$$\bar{h}(i) = h_2(i) - K_{21}(i)K_{11}(i)^{-1}h_1(i) ,$$

$$\bar{K}(i) = K_{22}(i) - K_{21}(i)K_{11}(i)^{-1}K_{12}(i).$$

The result follows.

When calculating marginals over discrete variables, there are two distinct cases. Firstly, consider the case where $h(i\,j)$ and $K(i\,j)$ do not depend on j. Then the canonical characteristics of the marginal distribution are given by

$$\tilde{g}(i) = \log \sum_{j:\chi(i,j)=1} \exp\,(g(i,j)),$$

$$\tilde{h}(i) = h(i,j), \qquad \tilde{K}(i) = K(i,j).$$

This can be shown by considering the CG potential below

$$\phi(i,j,y) = \chi(i\,j)\exp\,(g(i\,j) + h(i\,j)^T\,y - \frac{1}{2}y^T\,K(i\,j)\,y\,),$$

we want

$$\sum_j \phi(i\,j\,y) = \sum_j \chi(i\,j)\exp\,(\,g(i\,j) + h(i\,j)^T\,y - \frac{1}{2}y^T\,K(i\,j)\,y\,)$$

$$= \exp\left\{ \log \sum_{j:\chi(i\,j)=1} \exp\,(\,g(i\,j) + h(i\,j)^T\,y - \frac{1}{2}\,y^T\,K(i\,j)\,y\,) \right\}.$$

Therefore the canonical characteristics are as quoted above. If either of $h(i\,j)$ or $K(i\,j)$ depends on j then the marginalisation process is more complicated and is best considered by looking at the moment characteristics $(p\,,\xi,\Sigma)$. Firstly, consider the marginal probability $\tilde{p}(i)$. We have

$$p(i\,j) = p(X_{\Delta_1} = i\,, X_{\Delta_2} = j).$$

We want the marginal probability $\tilde{p}(i) = p\,(X_{\Delta_1} = i\,)$

$$p\,(X_{\Delta_1} = i\,) = \sum_{all\ j} p\,(X_{\Delta_1} = i\,, X_{\Delta_2} = j) = \sum_j p\,(i\,j)\,.$$

Consider next, the mean of the marginal distribution, $\xi(i)$. Using the standard result,

$$\mathbf{E}(Y \mid I = i) = \mathbf{E}(\,\mathbf{E}\,(Y \mid (I\,,J)) \mid I = i\,),$$

we obtain the expression,

$$\xi(i) = \sum_j \xi(i\,j)\,\frac{p(i\,j)}{p(i)}\,,$$

where $\frac{p(i\,j)}{p(i)}$ is the new normalised probability of occurrence for a particular $(i\,j)^{th}$ cell.

Lastly, consider the variance of the marginal distribution, the expression for this is obtained in a similar fashion to the expression for the mean, by making reference to another result on conditional expectations **E** and variances **V**,

$$\mathbf{V}(Y \mid I = i) = \mathbf{E}(\mathbf{V}(Y \mid (I, J)) \mid I = i) + \mathbf{V}(\mathbf{E}(Y \mid (I, J)) \mid I = i),$$

the expression for $\tilde{\Sigma}(i)$ follows directly from the above and is,

$$\tilde{\Sigma}(i) = \sum_j \Sigma(i\,j)\,\frac{p(i\,j)}{p(i)} + \sum_j (\xi(i\,j) - \xi(i))^T (\xi(i\,j) - \xi(i))\,\frac{p(i\,j)}{\tilde{p}(i)}\,.$$

When marginalising over both continuous and discrete variables we first marginalise over the continuous variables and then over the discrete.

Enter Evidence

We consider the operation of entering continuous evidence y^*_A on A to the following CG potential

$$\phi(i, x) = exp\left(g(i) + h(i)^T x - \frac{1}{2} x^T K(i)\, x \right).$$

The transformed potential $\phi^*(i, y)$ will have canonical characteristics $(g^*(i), h^*(i), K^*(i))$ given as

$$g^*(i) = g(i) + h_A(i) y^*_A - \frac{1}{2} K_{AA}(i) {y^*_A}^2,$$

$$h^*(i) = h_1(i) - y^*_A K_{A1}(i), \qquad K^*(i) = K_{11}(i).$$

PROOF:

If we partition the underlying space of the above CG potential into

$$x = \begin{bmatrix} y \\ y_A \end{bmatrix}, \quad h(i) = \begin{bmatrix} h_1(i) \\ h_A(i) \end{bmatrix}, \quad K(i) = \begin{bmatrix} K_{11}(i) & K_{1A}(i) \\ K_{A1}(i) & K_{AA}(i) \end{bmatrix},$$

then the new CG potential $\phi^*(i, y)$ will have the following distribution

$$\phi^*(i, y) = exp\left(g^*(i) + h^*(i)^T y - \frac{1}{2} y^T K^*(i)\, y \right).$$

The CG potential will be

$$\phi(i, x) = exp\left(g(i) + \begin{bmatrix} h_1(i) \\ h_A(i) \end{bmatrix}^T \begin{bmatrix} y \\ y_A \end{bmatrix} - \frac{1}{2} \begin{bmatrix} y \\ y_A \end{bmatrix}^T \begin{bmatrix} K_{11}(i) & K_{1A}(i) \\ K_{A1}(i) & K_{AA}(i) \end{bmatrix} \begin{bmatrix} y \\ y_A \end{bmatrix} \right)$$

$$= exp\left(g(i) + h_1(i)^T y + h_A(i)^T y_A - \frac{1}{2} \left(y^T K_{11}(i)\, y + y_A\, K_{A1}(i)\, y \right. \right.$$

$$+ y^T K_{1A}(i)\, y_A + (y_A)^2 K_{AA}(i)))$$

$$= exp\,(g(i) + h_A(i)^T y_A - \frac{1}{2}(y_A)^2 K_{AA}(i) + (h_1(i) - y_A\, K_{A1}(i))^T y - \frac{1}{2} y^T K_{11}(i)\, y)\,.$$

Now fixing y_A at y^*_A we get

$$\phi^*(i, x) = exp\,(g(i) + h_A(i)^T y^*_A - \frac{1}{2}(y^*_A)^2 K_{AA}(i) + (h_1(i) - y^*_A\, K_{A1}(i))^T y - \frac{1}{2} y^T K_{11}(i)\, y)\,.$$

This can be regrouped since y^*_A is now a constant and not a variable. The transformed potential $\phi^*(i, x)$ will have canonical characteristics $(g^*(i), h^*(i), K^*(i))$, where

$$g^*(i) = g(i) + h_A(i) y^*_A - \frac{1}{2} K_{AA}(i) {y^*_A}^2,$$

$$h^*(i) = h_1(i) - y^*_A K_{A1}(i), \qquad K^*(i) = K_{11}(i)\,.$$

These are the new canonical characteristics as quoted above.

Appendix D

MEDICAL DATABASE

This appendix lists all the diagnostic groups and the symptoms with their different values in the abdominal pain diseases database. The description of the database is then presented. Of the preliminary diagnoses made by doctors, an estimated accuracy is also given.

List of Diagnostic Groups

Diagnostic Groups		
Group No.	Diagnosis	Abbrev.
1	Appendicitis	APP
2	Diverticulitis	DIV
3	Perforated Peptic Ulcer	PPU
4	Non-specific Abdominal Pain*	NAP
5	Cholisistitis	CHO
6	Intestinal Obstruction	INO
7	Pancreatitis	PAN
8	Renal Colic	RCO
9	Dyspepsia	DYS

*NAP consists of all those patients who have not been diagnosed to one of the other diagnostic groups.

List of Symptoms with Their Values

Symptom	Values (No.)	Code
1.Sex	male(0),female(1)	1—2
2.Age	0-9(0),10-19(1),20-29(2),30-39(3) 40-49(4),50-59(5),60-69(6),70+(7)	3—10
3.Pain-site Onset	right upper quadrant(0),left upper quadrant(1),right lower quadrant(2) left lower quadrant(3),upper half(4),lower half(5),right half(6),left half(7) central(8),general(9),right loin(10),left loin(11),epigastric(12)	11—23
4.Pain-site Present	right upper quadrant(0),left upper quadrant(1),right lower quadrant(2) left lower quadrant(3),upper half(4),lower half(5),right half(6),left half(7) central(8),general(9),right loin(10),left loin(11), epigastric(12),pain settled(13)	24—37
5.Aggravating Factors	movement(0),coughing(1),inspiration(2),food(3),other(4),nil(5)	38—43
6.Relieving Factors	lying still(0),vomiting(1),antacids(2),milk/food(3),other(4),nil(5)	44—49
7.Progress of Pain	getting better(0),no change(1),getting worse(2)	50—52
8.Duration of Pain	under 12 hours(0),12-24 hours(1),24-48 hours(2),over 48 hours(3)	53—56
9.Type of Pain	steady(0),intermittent(1),colicky(2),sharp(3)	57—60
10.Severity of Pain	moderate(0),severe(1)	61—62
11.Nausea	nausea present(0),no nausea(1)	63—64
12.Vomiting	present(0),no vomiting(1)	65—66
13.Anorexia	present(0),normal appetite(1)	67—68
14.Indigestion	history of dyspepsia(0),no history of dyspepsia(1)	69—70
15.Jaundice	history jaundice(0),no history of jaundice(1)	71—72
16.Bowel Habit	no change(0),constipated(1),diarrhoea(2),blood(3),mucous(4)	73—77
17.Micturation	normal(0),frequent(1),dysuria(2),haematuria(3),dark urine(4)	78—82
18.Previous Pain	similar pain before(0),no similar pain before(1)	83—84
19.Previous Surgery	yes(0),none(1)	85—86
20.Drugs	being taken(0),not being taken(1)	87—88
21.Mood	normal(0),distressed(1),anxious(2)	89—91
22.Colour	normal(0),pale(1),flushed(2),jaundiced(3),cyanosed(4)	92—96
23.Abdominal Movements	normal(0),poor/nil(1),visible peristalsis(2)	97—99
24.Abdominal Scar	present(0),absent(1)	100—101
25.Abdominal Distension	present(0),absent(1)	102—103
26.Site of Tenderness	right upper quadrant(0),left upper quadrant(1),right lower quadrant(2) left lower quadrant(3),upper half(4),lower half(5),right half(6),left half(7) central(8),general(9),right loin(10),left loin(11),epigastric(12),none(13)	104—117
27.Rebound	present(0),absent(1)	118—119
28.Guarding	present(0),absent(1)	120—121
29.Rigidity	present(0),absent(1)	122—123
30.Abdominal Masses	present(0),absent(1)	124—125
31.Murphy's Test	positive(0),negative (1)	126—127
32.Bowel Sounds	normal(0),decreased(1),increased(2)	128—130
33.Rectal Examination	tender left side(0),tender right side(1),generally tender(2) mass felt(3),normal(4)	131—135

The Description of the Database

6387 patients with abdominal pain were collected by Dr. Gunn at Bangour Hospital, Livingston and were passed on by Dr. Nixon of West General Hospital, Edinburgh. Each patient was noted down his/her diagnosis made by a doctor, a set of observed symptoms with their values (states). These patients are classified into 9 diagnostic groups. It is assumed that a patient belongs to one and only one diagnostic

group. The number of patients in each diagnostic group and a sample of patient records are given as follows.

The Number of Patients in Each Diagnostic Group								
APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS
844	143	130	2835	572	417	96	473	877

A Sample of Patient Records		
Patient No.	Diagnosis	Symptoms (Code)
1	D=9	1 6 19 28 38 49 52 53 57 60 62 63 65 67 70 72 73 78 84 86 88 91 92 97 101 103 111 115 119 121 123 125 127 128 135
2	D=4	1 4 19 26 39 49 50 55 60 61 64 65 67 70 72 73 80 83 86 87 89 94 97 101 103 106 119 121 123 125 127 128 135
3	D=8	2 7 17 21 30 34 43 49 52 56 57 60 62 63 65 67 70 72 73 83 85 87 92 97 100 103 110 114 119 121 123 125 126 128 135
4	D=4	2 4 16 29 39 40 42 44 50 53 60 61 64 66 68 70 72 73 78 84 86 87 89 93 97 101 103 109 119 123 125 127 129 135
5	D=5	2 9 11 24 38 39 40 44 51 55 57 59 62 63 65 67 70 72 73 79 83 86 87 91 92 97 103 104 118 121 123 125 126 128 135
6	D=4	2 4 16 33 38 49 52 56 58 62 63 65 68 70 72 74 79 82 84 86 88 89 92 97 100 103 117 119 121 123 125 127 128 135

Actually, there are 135 symptoms/values taken into consideration. They include patient demographics (age, sex, etc.) and medical history, as well as signs and lab findings. For example, considering the first patient's record,

```
9   1   6   19   28  ...  103  111  115  119  121 ...
```

The first number indicates the patient's final diagnosis. The other numbers are the observed symptom values in their codes. By checking code in the table, the real meaning for S_i (i=1,2, ... , 135) can be found. Accordingly, the above patient record can be interpreted as:

Final Diagnosis: Dyspepsia

Symptoms:
Sex --- Male (1)
Age --- between 30 and 39 (6)
Pain Site Onset --- central (19)
Pain Site Present --- left lower quadrant (28)
.....
Abdominal Distension --- absent (103)
Site of Tenderness --- left half (111)
Site of Tenderness --- left loin (115)
Rebound --- absent (119)
Guarding --- absent (121)
.....

For computational convenience, a new coding system is used. Under the new coding system, the accumulated data can be represented by the following table,

Diagnosis	Symptoms			
$D^{(1)}$	$S_1^{(1)}$	$S_2^{(1)}$	...	$S_{33}^{(1)}$
$D^{(2)}$	$S_1^{(2)}$	$S_2^{(2)}$	...	$S_{33}^{(2)}$
...	...	...	...	...
$D^{(6387)}$	$S_1^{(6387)}$	$S_2^{(6387)}$	...	$S_{33}^{(6387)}$

where $D^{(i)} \in \{D_1, D_2, ..., D_9\}$, $S_j^{(i)} \in \{S_1, S_2, ..., S_{33}\}$. The value of each symptom is represented by a number within the range from 0 to the total number of specific attribute values minus 1, e.g. symptom vomiting has 2 values, present and no vomiting, they are represented by 0 and 1 respectively. In addition, two more values are used in the database which are '88' for multiple observations and '99' for missing observations. The patient record described above can be expressed as follows using the new coding system

```
9   0   3   8   3  ...  1  88  1  1 ...
```

Doctor's Diagnoses

It is interesting to know the accuracy of diagnoses made by doctors. The table below shows the results of 3,847 preliminary diagnoses by doctors at Bangour hospital (reproduced with permission of Mr. S. Nixon and Mr. A. Gunn). Of the preliminary diagnoses made by doctors on these diseases, an estimated 76.35 per cent are correct in the sense that they agree with the final diagnoses.

Doctor's Diagnoses vs. Final Diagnoses										
Doctor's Diagnoses	Final Diagnoses									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	430	2	2	49	6	7	4	2	4	506
DIV	4	56	3	16	1	16	0	1	3	100
PPU	4	1	60	8	6	2	1	1	3	86
NAP	299	12	7	1169	9	48	7	49	30	1630
CHO	3	1	12	11	256	12	11	2	33	341
INO	8	1	2	24	4	222	1	1	8	271
PAN	0	0	3	4	8	3	33	0	8	59
RCO	5	1	1	8	2	1	0	268	1	287
DYS	6	5	11	35	43	9	12	3	443	567
Total	759	79	101	1324	335	320	69	327	533	3847

Overall Accuracy=2937/3847=76.35%

Appendix E

PARAMETER LEARNING

This appendix presents the results of the experiments of the sequential updating technique on the large medical database (6387 patient records). As described in section 5.2.2, two systems, namely System 1 and System 2, were implemented using PRESS to carry out these experiments. System 1 and System 2 were initialised on the same training set and tested with the same remaining patient testing set. The two systems were evaluated on a set of data, which were randomly selected from the medical database. The training size was chosen as one of 100, 200, 300, 400, 500, 1000, 2000, 3000, 4000 and 5000 respectively. The remaining set was used as the testing set.

For each testing set, a table was set up to evaluate the performance of the systems by comparing the computer diagnoses with their final diagnoses made by the doctors. Three experiments were made on the same size of the different training sets. Therefore, 30 tables were made based on the experiments. In some cases, the system could not classify all the patient records in testing set due to small size of the training set. These patient records were referred as *unclassified cases*.

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 100)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	110	0	0	571	0	3	0	2	6	692
DIV	0	0	0	65	0	14	0	0	1	80
PPU	1	0	0	16	1	12	0	2	6	38
NAP	90	0	0	1873	6	24	0	22	100	2115
CHO	1	0	0	44	38	8	0	1	46	138
INO	1	0	0	150	0	51	0	1	13	216
PAN	1	0	0	8	2	5	0	0	14	30
RCO	0	0	0	123	1	11	0	79	7	221
DYS	2	0	0	79	25	9	0	2	305	422
Total	206	0	0	2929	73	137	0	109	498	3952

Overall Accuracy=2456/6287=39.06% Unclassified Cases: 2335

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 100)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	255	0	0	425	1	3	0	0	3	687
DIV	2	0	0	84	0	1	0	0	1	88
PPU	8	0	0	10	0	1	0	0	2	21
NAP	167	0	0	1925	23	5	0	1	15	2136
CHO	0	0	0	46	108	0	0	0	36	190
INO	8	0	0	191	6	7	0	0	6	218
PAN	1	0	0	26	5	0	0	0	10	42
RCO	7	0	0	142	13	2	0	5	6	175
DYS	7	0	0	165	38	1	0	0	164	375
Total	455	0	0	3014	194	20	0	6	243	3932

Overall Accuracy=2464/6287=39.19% Unclassified Cases: 2355

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 100)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	162	0	0	398	0	0	0	0	9	569
DIV	1	0	0	50	0	0	0	0	5	56
PPU	4	0	0	5	3	0	0	0	9	21
NAP	84	0	0	1931	8	1	0	4	68	2096
CHO	0	0	0	26	22	0	0	0	169	217
INO	2	0	0	157	0	1	0	0	18	178
PAN	0	0	0	8	2	0	0	0	31	41
RCO	1	0	0	99	6	0	0	10	13	129
DYS	2	0	0	96	9	0	0	0	383	490
Total	256	0	0	2770	50	2	0	14	705	3797

Overall Accuracy=2509/6287=39.91% Unclassified Cases: 2490

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 100)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	578	2	12	192	3	18	4	3	16	828
DIV	4	68	5	28	0	26	1	5	2	139
PPU	4	2	65	8	15	10	11	3	9	127
NAP	351	54	8	2051	49	90	1	84	96	2784
CHO	3	2	12	24	371	31	10	6	93	552
INO	14	23	11	50	12	255	8	6	22	401
PAN	2	2	5	4	17	12	19	1	33	95
RCO	7	15	3	62	11	6	1	335	13	453
DYS	12	6	17	74	65	27	26	6	621	854
Total	975	174	138	2493	543	475	81	449	905	6233

Overall Accuracy=4363/6287=69.40% Unclassified Cases: 54

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 100)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	597	2	11	173	4	18	3	3	17	828
DIV	4	65	4	30	0	26	1	4	2	136
PPU	7	2	65	5	15	6	12	2	11	125
NAP	361	56	9	2031	54	90	2	84	89	2776
CHO	3	2	13	23	372	28	11	6	96	554
INO	15	23	10	50	13	256	8	8	22	405
PAN	2	2	7	4	18	12	20	1	29	95
RCO	11	16	5	66	13	5	1	330	14	461
DYS	13	6	16	80	63	27	27	5	622	859
Total	1013	174	140	2462	552	468	85	443	902	6239

Overall Accuracy=4358/6287=69.32% Unclassified Cases: 48

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 100)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	578	3	12	180	4	19	3	3	16	818
DIV	4	63	5	30	1	27	1	5	2	138
PPU	6	2	63	4	15	7	13	3	15	128
NAP	355	56	9	2009	52	93	2	86	88	2750
CHO	3	2	12	21	367	29	11	7	105	557
INO	15	22	10	52	11	255	8	7	24	404
PAN	2	2	5	4	17	13	20	1	30	94
RCO	10	15	5	64	11	4	1	332	14	456
DYS	12	6	16	77	61	28	26	5	620	851
Total	985	171	137	2441	539	475	85	449	914	6196

Overall Accuracy=4307/6287=68.51% Unclassified Cases: 91

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 200)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	428	0	1	289	2	12	0	0	21	753
DIV	3	9	0	75	0	21	0	0	5	113
PPU	5	1	3	32	0	13	0	3	44	101
NAP	290	10	0	1874	60	35	0	8	141	2418
CHO	11	3	1	20	244	11	0	5	160	455
INO	13	3	0	135	8	86	0	1	52	298
PAN	2	0	2	9	6	5	0	0	53	77
RCO	9	5	0	222	10	7	0	38	24	315
DYS	16	1	1	101	57	7	0	2	577	762
Total	777	32	8	2757	387	197	0	57	1077	5292

Overall Accuracy=3259/6187=52.67% Unclassified Cases: 895

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 200)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	217	1	0	519	5	1	1	1	10	755
DIV	0	1	0	100	3	2	1	3	1	111
PPU	2	0	2	17	30	2	2	1	25	81
NAP	113	0	1	2241	40	7	1	45	53	2501
CHO	0	1	0	91	335	3	0	1	50	481
INO	5	4	0	268	25	20	0	3	22	347
PAN	0	1	0	13	40	0	2	0	23	79
RCO	2	1	0	125	18	0	1	152	7	306
DYS	6	3	0	161	240	1	3	0	277	691
Total	345	12	3	3535	736	36	11	206	468	5352

Overall Accuracy=3247/6187=52.48% Unclassified Cases: 835

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 200)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	250	0	0	523	0	3	1	2	20	799
DIV	4	0	0	98	0	6	0	3	10	121
PPU	8	0	0	68	0	1	0	2	37	116
NAP	149	0	0	2318	24	16	3	37	89	2636
CHO	2	0	0	105	180	2	7	2	176	474
INO	4	0	0	245	3	22	3	3	69	349
PAN	1	0	0	18	5	1	5	2	46	78
RCO	2	0	0	156	3	1	0	183	13	358
DYS	1	0	0	234	43	2	7	5	482	774
Total	421	0	0	3765	258	54	26	239	942	5705

Overall Accuracy=3440/6187=55.60% Unclassified Cases: 482

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 200)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	589	3	14	165	4	18	3	2	16	814
DIV	4	64	4	27	0	26	1	6	1	133
PPU	7	3	65	6	10	9	10	2	12	124
NAP	358	57	9	2007	53	94	2	82	96	2758
CHO	3	2	11	20	365	31	10	7	95	544
INO	15	26	11	43	12	254	8	6	25	400
PAN	2	2	6	3	15	10	21	0	34	93
RCO	11	18	6	65	9	7	0	332	12	460
DYS	11	5	16	74	62	27	23	7	617	842
Total	1000	180	142	2410	530	476	78	444	908	6168

Overall Accuracy=4314/6187=69.73% Unclassified Cases: 19

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 200)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	572	2	13	195	4	17	5	3	18	829
DIV	4	64	4	29	0	25	1	8	2	137
PPU	5	2	65	5	14	6	12	3	12	124
NAP	342	54	9	2010	49	85	2	84	92	2727
CHO	3	2	12	20	379	28	8	5	88	545
INO	14	22	11	50	13	255	10	7	23	405
PAN	1	1	6	4	16	11	21	0	30	90
RCO	10	13	6	57	13	4	0	332	12	447
DYS	11	6	15	77	70	26	28	5	615	853
Total	962	166	141	2447	558	457	87	447	892	6157

Overall Accuracy=4313/6187=69.71% Unclassified Cases: 30

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 200)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	567	4	12	196	3	17	3	3	18	823
DIV	4	66	5	29	0	26	1	8	2	141
PPU	7	2	65	5	14	6	12	3	12	126
NAP	339	54	10	2030	53	84	3	80	91	2744
CHO	3	2	12	23	382	26	9	3	89	549
INO	13	22	11	50	16	252	9	5	23	401
PAN	1	1	6	4	15	11	21	1	31	91
RCO	10	11	6	60	12	5	0	326	13	443
DYS	13	6	18	73	72	22	20	5	610	839
Total	957	168	145	2470	567	449	78	434	889	6157

Overall Accuracy=4319/6187=69.81% Unclassified Cases: 30

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 300)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	449	0	2	314	4	15	0	1	11	796
DIV	6	3	0	98	3	14	0	1	2	127
PPU	7	0	17	37	9	37	0	2	4	113
NAP	255	3	1	2211	43	28	0	38	78	2657
CHO	4	1	4	81	301	19	1	6	86	503
INO	22	4	2	201	17	75	0	5	34	360
PAN	3	1	3	18	33	9	0	0	18	85
RCO	13	1	0	142	16	9	1	189	18	389
DYS	8	2	8	197	128	13	3	4	461	824
Total	767	15	37	3299	554	219	5	246	712	5854

Overall Accuracy=3706/6087=60.88% Unclassified Cases: 233

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 300)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	495	0	5	245	6	14	1	1	7	774
DIV	4	0	96	0	15	1	5	3	124	
PPU	8	0	13	38	23	6	2	3	18	111
NAP	337	3	1	2028	61	84	1	54	60	2629
CHO	2	0	0	35	369	27	3	4	48	488
INO	15	0	0	150	27	137	0	12	8	349
PAN	1	0	1	11	36	7	6	1	16	79
RCO	17	0	1	115	13	5	1	186	19	357
DYS	9	0	3	123	209	57	2	8	365	776
Total	888	3	24	2841	744	352	17	274	544	5687

Overall Accuracy=3599/6087=59.13% Unclassified Cases: 400

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 300)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	402	0	0	355	1	7	2	2	17	786
DIV	3	11	0	55	2	33	0	7	7	118
PPU	8	0	1	40	12	10	9	1	38	119
NAP	209	9	0	2102	31	31	3	64	129	2578
CHO	3	0	0	61	285	5	3	3	142	502
INO	10	2	0	160	21	95	1	5	55	349
PAN	2	1	2	10	20	4	1	0	48	88
RCO	8	5	0	99	8	6	0	227	28	381
DYS	11	1	0	115	48	8	5	3	590	781
Total	656	29	3	2997	428	199	24	312	1054	5702

Overall Accuracy=3714/6087=61.01% Unclassified Cases: 385

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 300)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	575	5	11	173	3	19	2	3	17	808
DIV	4	65	4	33	0	24	1	4	2	137
PPU	7	2	65	6	12	9	9	2	7	119
NAP	344	57	9	1968	49	90	3	84	89	2693
CHO	2	2	13	19	372	30	9	7	89	543
INO	14	26	8	48	9	250	9	6	23	393
PAN	2	2	7	5	16	13	19	1	26	91
RCO	10	14	4	62	13	5	1	324	13	446
DYS	12	7	16	71	62	29	24	5	616	842
Total	970	180	137	2385	536	469	77	436	882	6072

Overall Accuracy=4254/6087=69.89% Unclassified cases: 15

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 300)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	581	4	13	152	3	20	2	3	16	794
DIV	4	67	4	30	0	26	1	4	2	138
PPU	4	2	63	5	16	7	10	2	11	120
NAP	360	51	10	1972	50	91	2	80	94	2710
CHO	3	2	12	20	374	29	11	6	88	545
INO	15	21	9	44	12	252	9	8	19	389
PAN	2	2	7	4	13	11	19	1	31	90
RCO	11	14	4	58	10	7	1	324	14	443
DYS	12	7	17	69	67	29	26	7	607	841
Total	992	170	139	2354	545	472	81	435	882	6070

Overall Accuracy=4259/6087=69.97% Unclassified cases: 17

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 300)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	566	1	12	180	2	18	6	3	17	805
DIV	4	67	4	24	1	27	1	5	2	135
PPU	4	2	65	5	12	6	14	3	14	125
NAP	343	58	9	1982	44	86	2	85	98	2707
CHO	3	1	14	21	368	29	8	6	92	542
INO	13	22	10	48	13	250	12	6	22	396
PAN	1	1	5	3	16	10	22	1	32	91
RCO	8	15	4	59	9	5	1	325	14	440
DYS	12	7	13	69	58	21	27	5	616	828
Total	954	174	136	2391	523	452	93	439	907	6069

Overall Accuracy=4261/6087=70.00% Unclassified cases: 18

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 400)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	470	4	0	279	5	7	2	0	10	777
DIV	2	38	1	62	3	11	1	4	0	122
PPU	14	6	18	18	21	1	9	1	22	110
NAP	228	27	7	2149	38	42	1	47	83	2622
CHO	7	3	5	84	297	20	6	0	80	502
INO	18	29	3	132	20	140	5	4	22	373
PAN	4	1	1	12	19	5	8	1	32	83
RCO	5	7	1	120	17	6	1	232	12	401
DYS	22	1	0	173	116	21	12	3	470	818
Total	770	116	36	3029	536	253	45	292	731	5808

Overall Accuracy=3822/5987=63.84% Unclassified Cases: 179

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 400)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	447	1	0	311	1	14	0	2	16	792
DIV	6	26	0	65	0	14	0	9	4	124
PPU	11	0	8	18	8	15	5	2	48	115
NAP	222	30	0	2125	37	28	7	75	92	2616
CHO	1	0	0	58	282	15	7	3	129	495
INO	16	14	0	169	15	126	1	9	28	378
PAN	1	0	0	9	18	6	7	2	39	82
RCO	5	5	0	104	3	8	4	263	16	408
DYS	15	4	2	143	62	15	12	4	547	804
Total	724	80	10	3002	426	241	43	369	919	5814

Overall Accuracy=3831/5987=63.99% Unclassified Cases: 173

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 400)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	487	0	0	259	4	12	0	2	14	778
DIV	8	27	0	40	0	38	0	3	3	119
PPU	17	6	7	21	5	18	0	3	41	118
NAP	271	24	0	2077	35	77	0	60	84	2628
CHO	3	3	2	55	298	32	0	5	104	502
INO	18	23	0	111	17	182	0	2	15	368
PAN	3	2	2	11	16	9	0	1	44	88
RCO	8	12	0	107	7	11	0	231	27	403
DYS	10	3	3	119	72	25	0	5	556	793
Total	825	100	14	2800	454	404	0	312	888	5797

Overall Accuracy=3865/5987=64.56% Unclassified Cases: 190

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 400)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	556	4	13	174	2	16	4	3	15	787
DIV	3	66	3	29	0	23	2	3	1	130
PPU	3	4	64	5	12	8	9	1	13	119
NAP	329	58	9	1956	48	79	2	82	86	2649
CHO	3	2	11	25	365	28	10	6	85	535
INO	13	26	10	44	9	247	11	6	23	389
PAN	2	3	7	3	15	7	20	1	29	87
RCO	9	16	5	58	11	6	1	327	9	442
DYS	12	7	14	73	66	24	30	6	604	836
Total	930	186	136	2367	528	438	89	435	865	5974

Overall Accuracy=4205/5987=70.23% Unclassified Cases: 13

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 400)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	572	2	12	169	4	15	5	3	19	801
DIV	4	65	4	27	0	23	1	5	1	130
PPU	4	2	62	7	10	4	14	2	15	120
NAP	334	62	8	1951	48	76	2	80	90	2651
CHO	4	2	12	24	359	26	10	6	92	535
INO	12	25	9	46	11	250	9	7	20	389
PAN	2	2	6	4	14	10	19	1	27	85
RCO	6	14	5	55	8	7	1	327	15	438
DYS	12	6	16	78	58	25	24	5	601	825
Total	950	180	134	2361	512	436	85	436	880	5974

Overall Accuracy=4206/5987=70.25% Unclassified Cases: 13

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 400)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	561	3	11	175	5	17	3	3	15	793
DIV	3	63	4	25	0	29	1	4	1	130
PPU	3	2	67	7	9	9	11	2	12	122
NAP	343	58	10	1950	49	87	2	81	88	2668
CHO	3	1	16	17	352	32	8	6	90	525
INO	16	18	10	39	12	255	8	5	21	384
PAN	3	1	6	4	18	12	20	1	28	93
RCO	10	17	6	60	7	7	0	327	14	448
DYS	12	6	15	58	63	25	26	6	602	813
Total	954	169	145	2335	515	473	79	435	871	5976

Overall Accuracy=4197/5987=70.10% Unclassified Cases: 11

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	400	10	0	338	3	13	0	3	23	790
DIV	9	5	0	77	0	29	0	4	8	132
PPU	12	5	4	15	9	34	0	2	39	120
NAP	259	14	1	2059	38	59	1	58	134	2623
CHO	21	2	1	37	328	24	4	1	109	527
INO	29	20	0	112	17	172	0	1	35	386
PAN	4	1	1	8	17	9	3	0	46	89
RCO	27	2	0	104	19	7	0	245	24	428
DYS	22	2	1	88	59	20	3	3	594	792
Total	783	61	8	2838	490	367	11	317	1012	5887

Overall Accuracy=3810/5887=64.72% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	544	0	3	193	5	17	0	3	12	777
DIV	18	9	0	68	2	24	2	7	3	133
PPU	56	0	7	17	6	5	1	5	23	120
NAP	373	21	1	1939	38	69	1	43	139	2624
CHO	31	0	0	32	278	27	2	5	156	531
INO	37	4	2	80	14	207	1	12	23	380
PAN	8	1	1	2	15	9	3	1	49	89
RCO	67	8	0	105	10	5	1	221	20	437
DYS	32	1	3	69	41	28	8	3	611	796
Total	1166	44	17	2505	409	391	19	300	1036	5887

Overall Accuracy=3819/5887=64.87% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	552	0	2	179	5	13	0	1	17	769
DIV	15	13	0	57	0	41	0	1	6	133
PPU	35	0	7	13	22	9	0	2	35	123
NAP	350	10	0	1997	47	111	0	27	67	2609
CHO	16	0	1	37	350	32	0	1	96	533
INO	26	2	0	67	19	232	0	2	28	376
PAN	6	1	2	5	17	10	0	2	51	94
RCO	27	1	0	178	10	21	0	182	15	434
DYS	24	3	3	98	85	38	0	3	562	816
Total	1051	30	15	2631	555	507	0	221	877	5887

Overall Accuracy=3895/5887=66.16% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	550	7	13	177	2	17	3	3	19	791
DIV	3	65	4	29	0	22	1	5	3	132
PPU	5	3	65	4	11	8	8	3	12	119
NAP	321	51	8	1946	42	83	2	76	96	2625
CHO	2	2	13	22	353	25	10	5	96	528
INO	11	22	10	46	10	247	7	5	25	383
PAN	2	1	6	4	14	9	19	1	34	90
RCO	9	12	3	59	10	6	1	314	13	427
DYS	12	7	14	64	60	20	19	7	589	792
Total	915	170	136	2351	502	437	70	419	887	5887

Overall Accuracy=4148/5887=70.46% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	561	2	11	163	4	16	3	3	15	778
DIV	3	61	4	30	0	25	2	3	5	133
PPU	6	2	62	6	10	7	12	3	11	119
NAP	341	53	8	1918	46	85	2	76	97	2626
CHO	3	2	12	16	355	28	8	5	103	532
INO	13	19	10	42	9	250	8	7	19	377
PAN	3	1	7	2	14	11	19	1	34	92
RCO	9	13	4	64	5	6	1	318	14	434
DYS	13	7	15	66	50	24	29	6	586	796
Total	952	160	133	2307	493	452	84	422	884	5887

Overall Accuracy=4130/5887=70.15% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 500)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	563	1	10	152	5	18	2	2	17	770
DIV	4	65	4	22	0	29	1	5	2	132
PPU	8	2	65	5	13	9	7	2	14	125
NAP	342	52	9	1905	47	96	2	70	86	2609
CHO	3	1	10	20	368	32	8	6	86	534
INO	13	19	10	38	8	253	8	5	19	373
PAN	2	2	8	4	14	9	18	1	37	95
RCO	8	15	4	68	9	6	1	311	11	433
DYS	12	5	16	75	63	26	23	5	591	816
Total	955	162	136	2289	527	478	70	407	863	5887

Overall Accuracy=4139/5887=70.31% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 1000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	498	5	1	168	5	22	1	1	13	714
DIV	4	36	1	32	0	35	2	2	1	113
PPU	12	3	30	7	17	17	8	2	15	111
NAP	314	22	1	1802	44	86	6	61	86	2422
CHO	2	2	8	22	319	25	14	3	72	467
INO	13	24	1	46	16	224	1	2	19	346
PAN	0	1	3	2	18	14	13	0	32	83
RCO	7	10	3	70	15	19	1	261	15	401
DYS	9	2	12	54	61	28	19	8	530	723
Total	859	105	60	2203	495	470	65	340	783	5380

Overall Accuracy=3712/5387=68.91% Unclassified Cases: 7

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 1000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	495	0	7	175	4	16	1	0	14	712
DIV	4	39	3	30	2	25	2	1	3	109
PPU	9	0	50	5	15	11	5	2	15	112
NAP	308	50	4	1816	35	59	3	49	55	2379
CHO	2	3	8	40	316	22	3	0	82	476
INO	13	26	7	60	23	200	0	5	23	357
PAN	2	1	3	5	21	11	6	0	34	83
RCO	6	16	5	82	15	12	0	269	7	412
DYS	10	7	9	96	53	16	13	3	533	740
Total	849	142	96	2309	484	372	33	329	766	5380

Overall Accuracy=3724/5387=69.13% Unclassified Cases: 7

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 1000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	523	2	5	143	4	16	4	1	13	711
DIV	5	40	1	34	0	36	2	4	2	124
PPU	8	2	42	7	14	16	12	1	9	111
NAP	310	33	4	1816	36	77	3	67	67	2413
CHO	2	1	6	29	307	23	8	3	87	466
INO	14	17	4	61	10	204	7	4	22	343
PAN	2	0	1	5	18	15	21	4	20	86
RCO	10	9	4	53	10	14	1	285	10	396
DYS	10	4	6	75	59	22	29	8	518	731
Total	884	108	73	2223	458	423	87	377	748	5381

Overall Accuracy=3756/5387=69.72% Unclassified Cases: 6

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 1000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	524	1	14	138	4	18	3	1	13	716
DIV	4	60	2	21	0	22	1	3	1	114
PPU	3	2	61	5	10	8	11	2	10	112
NAP	320	49	10	1774	39	83	3	69	80	2427
CHO	3	2	12	16	323	25	9	5	77	472
INO	11	25	6	38	10	226	10	3	18	347
PAN	0	1	7	2	12	9	22	0	26	79
RCO	8	14	6	50	10	6	0	292	12	398
DYS	10	5	13	58	49	22	27	7	531	722
Total	883	159	131	2102	457	419	86	382	768	5387

Overall Accuracy=3818/5387=70.78% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 1000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	522	1	16	131	4	14	3	3	18	712
DIV	3	59	5	15	0	27	1	3	2	115
PPU	2	2	72	3	10	3	8	2	10	112
NAP	319	57	9	1750	37	64	4	72	70	2382
CHO	3	0	11	21	324	27	6	5	76	473
INO	10	23	10	40	12	228	9	5	20	357
PAN	0	1	7	3	14	9	20	0	27	81
RCO	7	13	5	56	8	6	0	307	12	414
DYS	10	7	15	68	45	20	22	5	549	741
Total	876	163	150	2087	454	398	73	402	784	5387

Overall Accuracy=3831/5387=71.11% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 1000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	522	2	15	133	3	17	5	3	15	715
DIV	5	63	3	20	0	26	2	4	2	125
PPU	3	2	64	5	8	6	9	2	12	111
NAP	318	51	10	1767	36	80	2	69	81	2414
CHO	3	2	10	22	322	23	7	2	78	469
INO	13	24	7	38	8	224	6	4	21	345
PAN	2	1	6	4	14	9	20	1	25	82
RCO	8	12	3	54	8	5	1	289	13	393
DYS	10	7	13	57	49	21	28	6	542	733
Total	884	164	131	2100	448	411	80	380	789	5387

Overall Accuracy=3818/5387=70.78% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 2000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	425	2	6	141	3	13	2	1	16	609
DIV	1	47	2	24	0	14	1	2	2	93
PPU	2	2	54	3	8	5	7	1	3	85
NAP	224	46	6	1457	26	53	2	44	61	1919
CHO	1	0	4	17	264	24	5	2	79	396
INO	10	21	5	45	6	177	2	3	22	291
PAN	2	2	4	4	14	7	10	0	32	75
RCO	3	9	2	55	6	6	0	231	15	327
DYS	6	2	19	55	43	13	6	2	446	592
Total	674	131	102	1801	370	312	35	286	676	4387

Overall Accuracy=3111/4387=70.91% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 2000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	446	3	6	110	0	16	2	2	16	601
DIV	5	51	3	17	0	12	2	2	1	93
PPU	3	2	50	4	10	8	4	1	6	88
NAP	282	43	7	1449	28	61	4	56	48	1978
CHO	6	2	5	16	274	18	3	2	58	384
INO	6	25	5	31	8	177	7	6	18	283
PAN	2	1	2	2	12	10	12	1	16	58
RCO	10	14	3	50	10	3	0	215	12	317
DYS	12	5	9	59	44	15	21	5	415	585
Total	772	146	90	1738	386	320	55	290	590	4387

Overall Accuracy=3089/4387=70.41% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 2000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	397	2	9	137	2	15	2	1	12	577
DIV	3	34	3	35	0	23	0	2	2	102
PPU	3	3	54	6	5	11	2	2	9	95
NAP	215	25	6	1428	27	66	1	49	73	1890
CHO	6	1	11	19	274	20	5	4	61	401
INO	11	16	5	31	12	190	2	4	17	288
PAN	2	1	5	4	12	10	11	0	28	73
RCO	7	5	3	58	10	6	1	248	12	350
DYS	8	5	15	55	44	12	15	2	455	611
Total	652	92	111	1773	386	353	39	312	669	4387

Overall Accuracy=3091/4387=70.46% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 2000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	441	0	12	119	2	15	3	2	15	609
DIV	1	54	2	17	0	15	1	2	1	93
PPU	2	2	56	2	6	5	6	2	4	85
NAP	253	37	9	1415	29	55	4	52	65	1919
CHO	2	0	7	13	266	24	8	3	73	396
INO	10	25	5	37	8	178	6	4	18	291
PAN	2	2	5	1	13	6	19	0	27	75
RCO	4	9	2	50	7	4	0	239	12	327
DYS	6	1	20	47	40	13	15	3	447	592
Total	721	130	118	1701	371	315	62	307	662	4387

Overall Accuracy=3115/4387=71.00% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 2000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	439	2	12	114	1	14	2	2	15	601
DIV	5	52	2	16	0	15	1	1	1	93
PPU	2	2	55	4	6	6	6	1	7	89
NAP	270	49	9	1447	28	61	2	61	50	1977
CHO	5	2	6	12	274	17	5	5	58	384
INO	6	24	7	29	8	180	6	4	19	283
PAN	1	1	4	3	10	8	16	0	15	58
RCO	7	10	5	46	9	3	0	229	8	317
DYS	10	5	12	53	39	17	23	5	421	585
Total	745	147	112	1724	375	321	61	308	594	4387

Overall Accuracy=3113/4387=70.96% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 2000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	408	1	13	119	3	13	5	2	13	577
DIV	3	51	3	22	0	17	1	3	2	102
PPU	3	2	58	3	7	6	9	2	5	95
NAP	240	36	6	1388	29	60	1	59	72	1891
CHO	3	1	11	15	270	23	8	5	64	400
INO	10	17	9	28	9	194	2	4	15	288
PAN	2	1	7	2	11	9	16	0	25	73
RCO	8	8	5	45	7	4	0	262	11	350
DYS	9	5	18	50	42	16	18	5	448	611
Total	686	122	130	1672	378	342	60	342	655	4387

Overall Accuracy=3095/4387=70.54% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 3000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	347	1	4	95	1	12	1	2	11	474
DIV	0	43	2	12	0	9	1	4	1	72
PPU	4	1	28	4	7	6	3	1	7	61
NAP	195	33	4	1087	31	40	2	44	44	1480
CHO	1	2	7	9	217	11	9	2	46	304
INO	6	26	1	23	6	135	1	3	15	216
PAN	1	2	5	2	6	5	11	0	21	53
RCO	4	10	2	30	3	1	0	194	7	251
DYS	6	3	9	42	38	13	8	4	353	476
Total	564	121	62	1304	309	232	36	254	505	3387

Overall Accuracy=2415/3387=71.30% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 3000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	302	1	7	106	3	10	1	1	11	442
DIV	6	43	3	10	0	7	1	1	1	72
PPU	2	2	46	3	8	7	4	1	6	79
NAP	170	36	1	1139	19	47	1	39	53	1505
CHO	1	1	7	12	213	13	4	3	48	302
INO	6	19	3	23	6	139	3	1	12	212
PAN	0	2	6	3	11	4	11	1	20	58
RCO	5	10	3	39	3	4	0	182	8	254
DYS	8	5	10	37	35	14	7	6	341	463
Total	500	119	86	1372	298	245	32	235	500	3387

Overall Accuracy=2416/3387=71.33% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 3000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	305	2	9	94	2	8	0	1	11	432
DIV	3	40	4	16	0	13	1	1	1	79
PPU	4	2	38	1	5	6	1	2	6	65
NAP	208	33	5	1140	22	45	0	46	51	1550
CHO	2	2	3	13	199	12	3	4	43	281
INO	5	20	7	22	8	145	0	2	11	220
PAN	0	0	3	1	10	9	12	0	17	52
RCO	7	5	0	35	3	2	1	196	9	258
DYS	8	4	9	41	35	12	15	3	323	450
Total	542	108	78	1363	284	252	33	255	472	3387

Overall Accuracy=2398/3387=70.80% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 3000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	348	1	6	92	1	11	2	1	12	474
DIV	0	44	2	11	0	10	1	4	0	72
PPU	3	1	34	5	3	4	4	2	5	61
NAP	194	30	6	1089	29	39	2	48	43	1480
CHO	2	1	8	7	215	12	9	2	48	304
INO	7	25	3	22	5	133	2	3	16	216
PAN	1	2	6	1	6	4	12	0	21	53
RCO	4	8	2	34	3	2	0	191	7	251
DYS	6	1	10	34	36	15	9	6	359	476
Total	565	113	77	1295	298	230	41	257	511	3387

Overall Accuracy=2425/3387=71.60% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 3000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	312	1	8	92	2	11	3	1	12	442
DIV	5	42	4	11	0	7	1	1	1	72
PPU	1	2	50	3	8	6	2	1	6	79
NAP	183	31	5	1125	19	51	1	39	51	1505
CHO	1	1	7	11	212	14	4	3	49	302
INO	6	15	5	21	4	141	5	1	14	212
PAN	0	2	6	3	8	4	13	1	21	58
RCO	6	9	4	35	3	4	0	186	7	254
DYS	8	4	11	35	35	13	13	5	339	463
Total	522	107	100	1336	291	251	42	238	500	3387

Overall Accuracy=2420/3387=71.45% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 3000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	300	2	11	96	2	7	2	2	10	432
DIV	3	42	4	14	0	11	1	3	1	79
PPU	2	2	42	1	4	5	1	2	6	65
NAP	204	38	5	1138	23	47	0	45	50	1550
CHO	1	2	5	12	201	11	5	4	40	281
INO	4	22	8	22	4	144	3	2	11	220
PAN	0	1	4	1	11	8	14	0	13	52
RCO	7	6	2	36	4	2	0	194	7	258
DYS	8	4	11	42	34	11	15	2	323	450
Total	529	119	92	1362	283	246	41	254	461	3387

Overall Accuracy=2398/3387=70.80% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 4000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	215	2	5	68	1	8	1	0	3	303
DIV	3	32	2	11	0	11	1	1	0	61
PPU	2	0	32	0	6	0	2	0	1	43
NAP	133	26	9	759	18	36	4	30	31	1046
CHO	0	0	5	10	153	13	4	4	31	220
INO	4	13	6	19	2	104	2	1	3	154
PAN	0	1	1	2	8	2	10	0	6	30
RCO	5	6	3	17	2	3	0	140	2	178
DYS	5	1	12	30	27	13	12	4	248	352
Total	367	81	75	916	217	190	36	180	325	2387

Overall Accuracy=1693/2387=70.93% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 4000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	243	0	3	62	1	6	0	2	8	325
DIV	1	29	3	10	0	6	0	0	1	50
PPU	3	1	36	1	3	3	1	0	2	50
NAP	132	26	4	761	17	25	3	23	51	1042
CHO	0	0	9	10	156	11	7	1	32	226
INO	5	10	6	13	6	109	2	2	12	165
PAN	1	0	2	0	5	4	6	0	12	30
RCO	1	6	1	25	4	3	2	130	6	178
DYS	5	2	7	20	26	10	14	3	234	321
Total	391	74	71	902	218	177	35	161	358	2387

Overall Accuracy=1704/2387=71.39% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 4000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	222	1	10	61	4	11	0	1	5	315
DIV	0	31	2	12	0	10	0	4	1	60
PPU	2	3	44	1	2	3	1	1	5	62
NAP	141	16	1	799	21	24	2	32	36	1072
CHO	4	1	4	7	157	9	5	0	25	212
INO	3	11	3	14	5	106	0	3	9	154
PAN	0	2	4	0	4	4	11	0	12	37
RCO	1	5	4	25	2	0	0	128	5	170
DYS	4	2	9	23	20	15	12	2	218	305
Total	377	72	81	942	215	182	31	171	316	2387

Overall Accuracy=1716/2387=71.89% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 4000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	214	1	7	67	0	9	2	0	3	303
DIV	3	34	2	11	0	9	1	1	0	61
PPU	1	0	33	0	6	0	1	0	2	43
NAP	130	25	8	767	16	35	1	32	32	1046
CHO	0	0	6	10	153	11	3	4	33	220
INO	5	14	6	18	2	102	1	1	5	154
PAN	0	0	1	1	9	3	10	0	6	30
RCO	5	5	3	16	2	3	0	142	2	178
DYS	5	2	11	27	23	12	14	5	253	352
Total	363	81	77	917	211	184	33	185	336	2387

Overall Accuracy=1708/2387=71.55% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 4000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	240	0	5	62	0	7	2	1	8	325
DIV	1	29	3	9	0	8	0	0	0	50
PPU	2	1	37	1	4	3	0	0	2	50
NAP	133	23	5	764	17	26	2	23	49	1042
CHO	1	0	8	10	156	12	7	1	31	226
INO	5	12	7	11	6	108	2	2	12	165
PAN	1	0	2	1	5	5	7	0	9	30
RCO	1	4	3	22	5	3	0	132	8	178
DYS	5	2	9	20	26	10	11	2	236	321
Total	389	71	79	900	219	182	31	161	355	2387

Overall Accuracy=1709/2387=71.60% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 4000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	224	2	10	59	1	9	2	1	7	315
DIV	0	30	2	12	0	11	0	4	1	60
PPU	1	2	44	2	2	2	2	1	6	62
NAP	143	17	1	799	19	26	2	34	31	1072
CHO	4	1	5	6	159	9	4	0	24	212
INO	3	10	4	14	4	108	0	2	9	154
PAN	0	1	6	0	5	5	9	0	11	37
RCO	2	4	3	25	2	0	0	128	6	170
DYS	4	4	7	22	19	14	13	2	220	305
Total	381	71	82	939	211	184	32	172	315	2387

Overall Accuracy=1721/2387=72.10% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 5000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	128	1	3	39	0	5	1	0	7	184
DIV	1	24	2	9	0	4	0	0	1	41
PPU	0	1	22	1	1	1	3	1	2	32
NAP	72	9	3	469	9	14	0	15	22	613
CHO	0	2	3	8	90	4	6	0	21	134
INO	2	8	3	7	2	54	1	0	7	84
PAN	0	0	3	1	4	0	8	0	6	22
RCO	1	2	1	11	1	1	0	72	1	90
DYS	4	1	5	8	16	6	1	2	144	187
Total	208	48	45	553	123	89	20	90	211	1387

Overall Accuracy=1011/1387=72.89% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 5000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	133	0	6	58	0	3	1	0	1	202
DIV	0	14	2	3	0	6	0	0	1	26
PPU	1	1	18	0	3	2	2	1	1	29
NAP	73	11	2	472	12	26	0	16	18	630
CHO	1	0	3	1	87	6	3	1	17	119
INO	4	8	2	8	0	48	3	1	5	79
PAN	0	0	2	1	4	0	6	0	9	22
RCO	4	9	2	6	4	1	0	74	3	103
DYS	4	2	4	9	13	6	7	1	131	177
Total	220	45	41	558	123	98	22	94	186	1387

Overall Accuracy=983/1387=70.87% Unclassified Cases: 0

Computer Diagnoses (System 1) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 5000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	136	0	2	39	1	3	1	2	2	186
DIV	0	14	1	5	0	4	0	2	0	26
PPU	0	0	16	3	0	1	0	2	0	22
NAP	83	8	4	450	11	14	0	14	26	610
CHO	1	0	3	4	96	7	6	2	17	136
INO	3	3	4	9	1	55	3	1	5	84
PAN	1	0	1	1	6	1	3	0	7	20
RCO	1	4	3	13	1	1	0	84	6	113
DYS	2	0	9	16	14	3	6	0	140	190
Total	227	29	43	540	130	89	19	107	203	1387

Overall Accuracy=994/1387=71.66% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 5000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	128	1	3	39	0	5	1	0	7	184
DIV	1	26	2	7	0	4	0	0	1	41
PPU	0	1	22	1	2	1	2	1	2	32
NAP	71	10	3	464	10	15	0	16	24	613
CHO	0	2	4	8	90	5	3	0	22	134
INO	2	9	3	6	2	54	1	0	7	84
PAN	0	0	4	1	4	0	7	0	6	22
RCO	1	2	1	11	1	0	0	72	2	90
DYS	4	1	5	9	14	7	1	2	144	187
Total	207	52	47	546	123	91	15	91	215	1387

Overall Accuracy=1007/1387=72.60% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 5000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	135	0	6	56	0	3	1	0	1	202
DIV	0	14	2	3	0	6	0	0	1	26
PPU	1	1	19	0	3	1	2	1	1	29
NAP	72	12	2	475	11	26	0	16	16	630
CHO	1	0	3	2	85	6	3	1	18	119
INO	4	7	3	8	0	48	3	1	5	79
PAN	0	0	2	1	4	0	6	0	9	22
RCO	4	9	2	6	4	0	0	75	3	103
DYS	4	2	4	8	13	6	7	1	132	177
Total	221	45	43	559	120	96	22	95	186	1387

Overall Accuracy=989/1387=71.30% Unclassified Cases: 0

Computer Diagnoses (System 2) vs. Final Diagnoses										
Final Diagnoses	Computer Diagnoses (Training Set Size: 5000)									
	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	137	0	2	38	1	3	1	2	2	186
DIV	0	13	1	6	0	4	0	2	0	26
PPU	0	0	16	3	0	1	0	2	0	22
NAP	85	8	4	447	12	14	0	15	25	610
CHO	1	0	3	4	97	7	4	2	18	136
INO	3	4	3	10	2	54	3	0	5	84
PAN	1	0	1	0	6	1	3	0	8	20
RCO	1	3	3	14	1	1	0	83	7	113
DYS	2	0	8	16	15	2	6	0	141	190
Total	230	28	41	538	134	87	17	106	206	1387

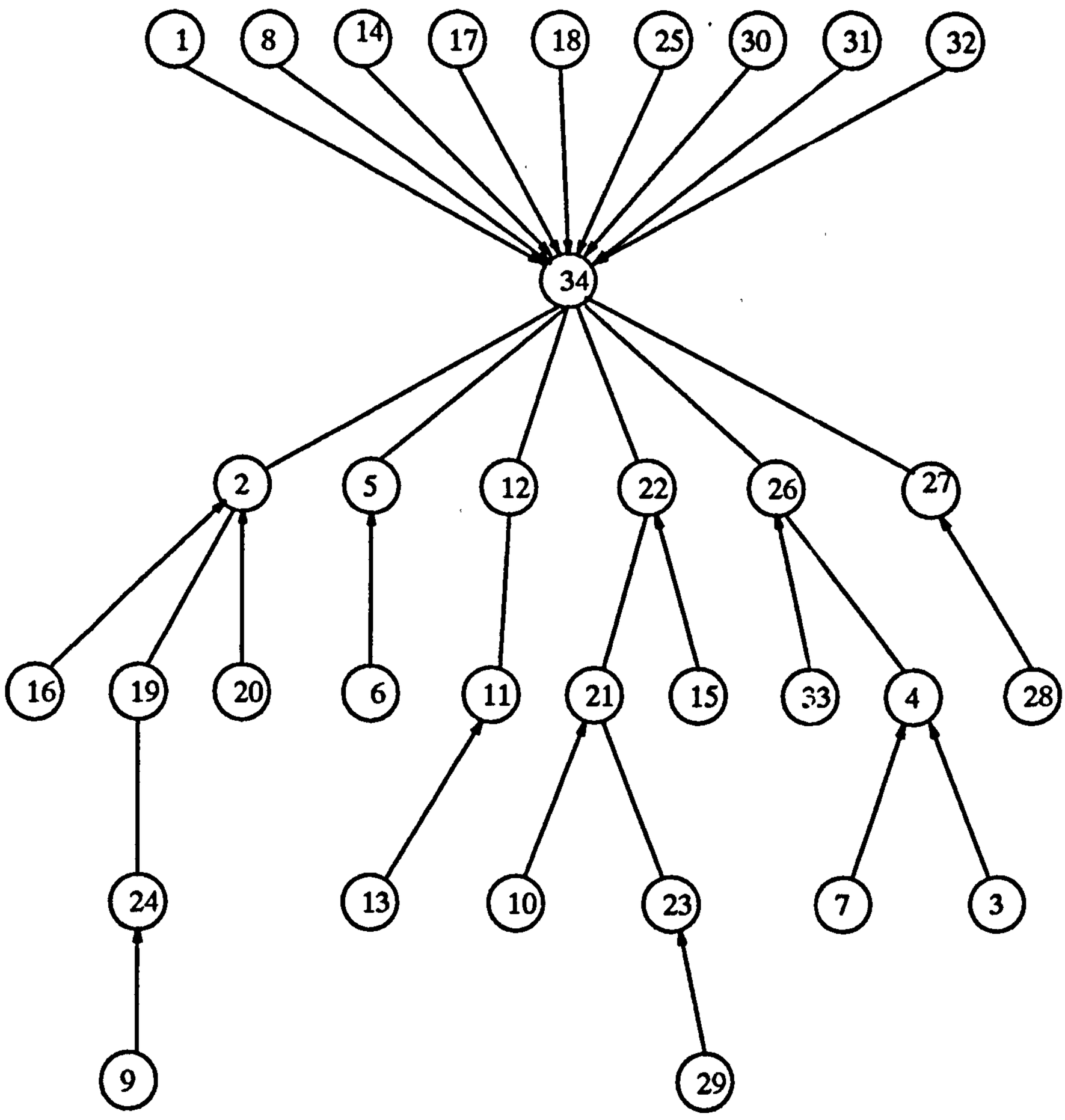
Overall Accuracy=991/1387=71.45% Unclassified Cases: 0

Appendix F

STRUCTURE LEARNING

This appendix presents a polytree constructed by Pearl's algorithm [29], which is described in section 5.3.1 and possible computer diagnoses obtained by using the skeleton tree.

Polytree Constructed by Algorithms 1 and 2



Computer Diagnoses Obtained by Using the Skeleton Tree

In order to make the diagnoses we calculate the posterior probabilities of each diagnostic group given the symptoms using the tree dependency structure (see Figure 5.7). So the calculations only include those symptoms that are directly connected with the root of the tree (No. 34). For example, for each patient of the testing set we compute:

$$P(d_1 \mid s_1, s_2, \ldots, s_{33}) = \alpha P(d_1)P(s_1 \mid d_1)P(s_2 \mid d_1)P(s_5 \mid d_1)P(s_8 \mid d_1)$$
$$P(s_{12} \mid d_1)P(s_{14} \mid d_1)P(s_{17} \mid d_1)P(s_{18} \mid d_1)P(s_{22} \mid d_1)P(s_{25} \mid d_1)$$
$$P(s_{26} \mid d_1)P(s_{27} \mid d_1)P(s_{30} \mid d_1)P(s_{31} \mid d_1)P(s_{32} \mid d_1),$$

$$\ldots\ldots\ldots\ldots\ldots$$

$$P(d_9 \mid s_1, s_2, \ldots, s_{33}) = \alpha P(d_9)P(s_1 \mid d_9)P(s_2 \mid d_9)P(s_5 \mid d_9)P(s_8 \mid d_9)$$
$$P(s_{12} \mid d_9)P(s_{14} \mid d_9)P(s_{17} \mid d_9)P(s_{18} \mid d_9)P(s_{22} \mid d_9)P(s_{25} \mid d_9)$$
$$P(s_{26} \mid d_9)P(s_{27} \mid d_9)P(s_{30} \mid d_9)P(s_{31} \mid d_9)P(s_{32} \mid d_9),$$

where α is a normalising constant, d_i (i=1,....,9) are 9 diagnostic groups and s_j (j=1, ..., 33) are values of corresponding symptoms. As a "computer diagnosis" we take the diagnostic group with the highest probability given the observed symptoms. Note that in doing so we assume that all 15 symptoms directly connected with the root are conditionally independent. The comparison between computer diagnoses and final diagnoses is given in the following table.

Computer Diagnoses vs Final Diagnoses										
	Computer Diagnoses									
Final Diagnoses	APP	DIV	PPU	NAP	CHO	INO	PAN	RCO	DYS	Total
APP	146	1	2	90	2	6	0	3	9	259
DIV	1	8	2	10	1	11	0	0	2	35
PPU	0	0	25	4	2	6	2	0	3	42
NAP	84	7	2	710	13	14	0	25	39	894
CHO	1	1	4	13	138	12	0	4	27	200
INO	3	5	1	19	1	83	0	2	13	127
PAN	2	0	2	0	4	3	2	1	17	31
RCO	2	0	0	36	8	5	0	86	10	147
DYS	3	2	5	48	26	10	1	6	164	265
Total	242	24	43	930	195	150	5	127	284	2000

Overall Accuracy=1362/2000=68.1%

Appendix G

SOFTWARE ASPECTS OF PRESS

PRESS is a computer program for probabilistic reasoning and learning in Bayesian belief networks in which the nodes represent random variables (discrete or continuous) whose conditional independence properties are captured in the topological structure of the directed acyclic graph (DAG). A graphical user interface programmed in SunView is provided. A copy of the program is available on the request.

Getting Started

To start PRESS, you must be in SunView environment, where you type press; and four windows should then appear. At this stage, PRESS has no model in memory, and awaits instructions to load one model from the directory call DATA or to construct one given by the user. Saving a model, propagation of probabilities, displaying of probability table held in a model can only performed if a model is active. A tree-structured menu system is used to interface the operations available in PRESS to the user. The main menu shown by PRESS has the following items:

- **Knowledge Acquisition**
- **Exact Algorithm**
- **Stochastic Simulation**
- **Display Probability**
- **Control Facilities**

where Knowledge Acquisition, Exact Algorithm, Stochastic Simulation and Control Facilities have their own submenus. Stochastic simulation and control facilities are implemented for pure discrete models. Different flags are used to indicate the operations chosen by the user.

Knowledge Acquisition

In general, the specification of such models requires the following information: (1) the topological nature of the graph representing independence of the variables (create model); a flag *mixed* is set if the model has both discrete and continuous nodes. (2) the specification of each conditional probability table in the model (input probability). There are two types of probability tables for discrete variables, the type is determined by the user selecting from the following menu

- Dirichlet
- Discrete

A flag *cpttype* is used to mark whether the conditional probability distributions specified are Dirichlet. Internally, point conditional tables are used in evidence propagation. We can convert the Dirichlet form of conditional probability tables - count tables - within the model by calling *freq2prob()*. Counts are interpreted as the elements $(\alpha_1, ..., \alpha_n)$ of a Dirichlet distribution for some conditional distribution of a node V_i and fixed parent configuration. The means $\alpha_i / \sum_j \alpha_j$ will be the conditional distribution for that parent configuration.

Exact Algorithm

The submenu for Exact Algorithm is

- Initialisation
- propagate evidence

Initialisation should always be called first when working in evidence propagation. If a clique tree is not set up, decomposition is required. The procedures *make_triangulated()*, *get_cliques()*, *max_card_search()*, *running_property()*, and *junction_tree()* are responsible for this. Operations *marry_parent()* and *fill_in()* are involved in the procedure *make_triangulated()*. Cliques, separators and the corresponding clique tree are worked out.

Initialisation sets all entries in the potential tables in the cliques and separators of the models to 1.0 and then multiplies the potentials in the cliques pointwise with the conditional probability tables held in the

model so that the clique tree has potentials which form a potential representation of the overall joint probability distribution (*pt_potential()*). All the evidence observed so far will be discarded. This option also allows the user to learn from a set of propagated evidence (**with learning**), i.e. to update the count tables by calling *caseupdate()*. The new point conditional tables are estimated. The learning is implemented for Dirichlet distributions.

Propagate Evidence allows the user to enter evidence selectively at one or more nodes and perform the L-S propagation algorithm. To enter evidence into a random variable means to select a specific state of the random variable. During propagation, two recursive procedures *collectevidence()* and *distributeevidence()* are called to perform the message passing algorithm in the clique tree *ctree*. **Propagate Evidence** can work in the **interactive** mode or **batch**-process data held in the file **EVIDENCE**. New marginal distributions are evaluated after propagation.

Stochastic Simulation

One of the uses of PRESS is to perform experiments to test learning algorithms. Using this option a Monte Carlo set of random complete data may be generated. Two possibilities are provided: **initialisation** and **propagate evidence.** The user is prompted for the total sample size to be generated. Output is written to the file **SAMPLE.** The current marginal probabilities are also estimated from these simulated values.

Control Facilities

This option allows the user to find the most influential evidence for a belief (**influential finding**) and to order questions concerning a node of interest (**planning**). An evidence propagation algorithm is used to perform planning. The computational algorithm is as follows: -

```
BEGIN
  specify a node of interest V_i
  calculate the current marginals P_c
  save the current information on the model M_c
  FOR each state j of V_i DO
  BEGIN
       propagate the impact of V_ij
       store the updated marginals P_ij
       reset the model to M_c
  END
  FOR each node V_k DO
       calculate Kullback-Leibler distances using P_c and P_ij
END
```

For influential finding, we save the updated marginals of nodes after each piece of evidence has been propagated (*prob_change*). The impact of evidence estimated by Kullback-Leibler distance is directly calculated from the *prob_change* (see section 3.2.2.2) and stored in *influencenode*.

Display Probability

If you want to know the current marginal probabilities of discrete nodes, means and variances of continuous nodes, **Display Probability** should be selected. The procedure *node_marginal()*, which marginalises the distributions on the clique, is called. For mixed models, both *continuous_marg()* and *discrete_marg()* procedures may be used to calculate marginals. You will then be shown a list of variable names and associated labels, marginal probabilities for discrete nodes. The observed nodes are not written on the screen. means, variances for continuous nodes.

Data Files in PRESS

The following table is a list of the files generated or used by PRESS.

SAMPLE	EVIDENCE	XXX.dat	sdump

The file SAMPLE contains the sample configurations generated by **Stochastic Simulation**, one sample per-line, which represent complete data. The file XXX.dat under the directory **DATA** contains the information of the models (both the structure and conditional probability tables). The file EVIDENCE contains a set of observed evidence for batch propagation. The file sdump created by *print* contains the

dump of screen of PRESS.

Data Structures

The main data structure is called a *d_node*. Inside the *d_node* reside the random variables, called *box_infos*. The maximum number of nodes at present is 100. Each *box_info* holds the name of a unique random variable, together with the names of a finite set of states if it is discrete, its parents index and its childrens index. The *box_infos* do not hold numerical values for probabilities. The probabilities are held in either the *node_disctables* (for discrete variables) or the *node_conctables* (for continuous variables). Each of them defines a family of random variables, through the set of their respective bases, for which conditional probability tables are required. A clique tree (*ctree*) of the associated DAG is made up of *potentials* and *separat_pts*, which are used for propagating probabilities and evidence. Both the *potentials* and the *separat_pts* are linked lists of structures. The pointers to the first structure in each list are stored globally after initialisation of the lists. For both lists, they are stored in a dynamically created structure (*new_cliqueprob*). The *potentials* hold configurations of the clique members and their potential representations. On the other hand, *separat_pts* hold information for the separators. Marginal tables (*node_margs*) store the current marginal values. The interpretation of these marginals will depend upon the type of random variables and propagation performed. The observed evidence is held in the components of *evidence*.

All data structures in the tabular form are represented by arrays; including an array of strings (used for the node names and two-dimensional arrays), the DAG, triangulated graph, clique table and stack. The number of nodes in the DAG and the number of cliques are represented globally. The procedure responsible for the input of node relationships requires the user to supply a list of node labels, separated by commas for each node of the graph. The same procedure is applied to the input of node values. The data structure used to record the conditional probabilities is a list of two-dimensional tables. The index system described in section 3.3.1 is used to read or write in the tables, with the first index representing the state of the node and the second index representing the mapping of its parent configuration.

Description of the Program

Main Functions	Main Procedures
Model Construction	setup_proc() get_condprobs() change_negations() read_distable() write_distable() read_contable() write_contable()
Preprocessor	make_triangulated() marry_parents() fill_in() max_card_search() get_cliques() running_property() junction_tree() initialisation() pt_potential()
Enter Evidence	get_evidence() absorb() continuous_enter() discrete_enter()
Evidence Propagation	propagation() collectevidence() distributeevidence() update()
Display Probability	node_marginal() node_distribution()
Control Facilities	planning() influence_find()
Stochastic Simulation	simulation() activatenodes()

Operations	Main Procedures
CG Potentials	expand() continuous_marg() discrete_marg() moment() shift()
Matrix	transpose() inverse() det() multiply() minus() plus()

setup_proc()

name the nodes in the DAG, get the possible value of each node, specify its parents and children

Consecutive positive integers are used as labels. The user is asked to associate a name with each consecutive label displayed on the screen. The value of the node is specified by the user and stored in the

table *nodes_value*. The number of values is stored for a discrete node and "-2" is stored for a continuous node. The relationships of the node are also determined.

get_condprobs()

get the conditional probability tables for each node in the graph either from the keyboard or from the data file and all of the conditional probability table may be recorded in the data file .

A conditional table must be supplied for each node, and for each combination of random variable values of the node's directed parents. The procedure *get_condmat* is designed for such purpose. For each node, the name of the node should be displayed, along with the list of conditions found in the graph table *cgraph*. The conditions should be displayed using a separate procedure which notes with the state of each condition and displays the corresponding name. A procedure *change_negation* can change the markers by searching down the corresponding row of the graph each time and the current node is to be displayed with another combination of conditions.

As each conditional probability is obtained, it should be placed in a data structure. An index system is used to store all the conditional probabilities of a node in the two-dimensional table *prob_table*. Given a certain combination of its parent nodes, an index is calculated and the corresponding probabilities can be easily accessed.

read_distable() write_distable() read_contable() write_contable()

read or write conditional probabilities from the corresponding table "prob_table"

Given a node and a certain combination of its parent nodes, an index is calculated using the same index system above. The proper probability values are read from the table or written to the corresponding table. These provide an efficient way of using conditional probabilities.

max_card_search()

perform a maximum cardinality search in the graph table "cgraph"

It can be determined from the table *graph* which node is discrete and a proper integer is assigned to *node_tally* of discrete nodes. Number 1 is given to any top most node(that is, the node does not have any parent nodes). Number the nodes consecutively, choosing as the next to number a node with the maximum number of *node_tally*. Break ties arbitrarily. *node_tally* for nodes connected to current numbered node is incremented. The simple algorithm will simultaneously check whether the graph is triangulated.

make_triangulated()

change the directed graph into an undirected graph and construct a triangulated graph; the undirected graph is stored in an array "tgraph"

The graph *cgraph* is initially processed by the procedure *marry_parents()*. The direction on the graph is then dropped. A test *fill_in()* to check whether an undirected graph *tgraph* is triangulated is carried out, if not, additional edges will be added to make it so.

marry_parents()

add undirected links between all co-parents that are currently unjoined

fill_in()

fill in sufficient additional links to ensure that there is no cycle of length 4 or more without a short-cut

An algorithm is implemented based on the paper "Algorithmic aspects of vertex elimination on graphs" by D.J. Rose, E. Tarjan and G.S. Lueker. SIAM J. COMPUT. Vol.13 No. 3 (1984) Selected a node based on MCS, the node to receive label *i* is that which has the most labelled neighbours. The stage is successful if those labelled neighbours of *i* are all neighbours of each other, i.e. form a complete subgraph.

get_cliques()

find the maximal cliques in the triangulated graph table "tgraph"

From the table, it can be seen which other nodes each of the nodes is connected to, so the node labels on

any row of the table, in conjunction with the corresponding control node, could form a clique. It is already known that the control node is connected to each of the other nodes in the row. It remains to be shown that each of the nodes in that row is connected to all other nodes in the row to verify that such a clique exists. If the nodes in the row do not all belong to one clique, a number of cliques may exist, each containing some subset of these nodes and the control node.

Thus far, the procedure has been concerned with sorting cliques which may be sub-cliques of those already stored or due to be stored. Only *maximal* cliques may be used, so once all the cliques have been stored, a procedure *sub_clique()* will be required to discard any sub-cliques. So, the cliques should be stored in a temporary data structure *temp_clique* initially and those which are maximal should be recorded in a new structure *clique*.

The cliques found can be stored in two-dimensional table *clique*, each row of which corresponds to the nodes comprising a clique. The number of the row will act as a unique identification label for the corresponding clique. The table should be associated with an integer *cliques* representing the number of cliques in the label.

running_property()

find a clique order which has the running-intersection property, in that the nodes of a clique also contained in previous cliques are all members of just one previous clique, known as a parent clique

A procedure *get_max_labels* is used to find maximum node label for each clique, associate clique label with each such node in the array "max_labels". Then we rank cliques according to the highest labelled node in each clique. Separator(*separator*), residual(*residual*) and possible parent cliques *parents_clique* are obtained for each clique. The separator is a set of intersection nodes between a clique and its previous cliques.

junction_tree()

construct a cliques tree from the triangulated graph tgraph

Each clique is a node in the cliques tree *ctree*. An edge is set up between a clique and one of its possible parent cliques. This ensures that communication from any node to another is via a unique path.

initialisation()

initialise the clique tree and potentials on cliques and separators are calculated from conditional probability tables and they are ready to propagate evidence

The clique marginals and separator marginals are generated and can be accessed by *potential* and *separat_pt*. Each of them is stored in a data structure which records a one-dimensional table of clique or separator nodes, with the appropriate markers to indicate the random variable values, and a potential. Therefore, one potential is required for each clique for each combination of random variable values of its nodes. Thus each pieces of information on the clique marginal is stored dynamically by *new_cliqueprob()*. Random variable values for each clique will have to be found using the same technique as that used to change the random variable values of conditions in the conditional probability elicitation section.

pt_potential()

calculate initial CG potentials on cliques from the conditional probability tables

The clique marginals are calculated from the conditional probability tables *node_disctable* and *node_conctable*. The CG potentials of mixed models in the triple (g,h,K) are stored in two data structures *potential* and *separat_pt*.

node_marginal()

find out marginal probabilities of discrete nodes, means and variances of continuous nodes

The marginal probabilities of discrete nodes, means and variances of continuous nodes can be obtained by using two procedures *discrete_marg()* and *continuous_marg()* on the clique containing the node of interest. The information is placed in the data structure *node_marg*, which will be accessed by the procedure *node_display* and presented to the user.

get_evidence()

ask the user to specify evidence for a variable in the graph

The procedure checks first that there are still some nodes which have not already been realised. Then it lists all those nodes. The user is asked to select one of the nodes displayed. Evidence is declared for the selected node. The state of the observed node is specified for a discrete node. A particular value is asked for from an observed continuous node.

absorb()

absorb observed evidence in a proper clique

Two procedures, *continuous_enter()* and *discrete_enter()*, are used for entering discrete evidence and continuous evidence, respectively. The CG potentials on these cliques are changed accordingly.

propagate()

propagate evidence in the clique tree

A message passing technique is used. A root clique is chosen arbitrarily. A procedure *collectevidence()* is carried out in the cliques tree, *tree*. Then the procedure *distributeevidence()* is carried out. At the end of this "distribution" phase, full equilibrium is reached. Each clique uses fundamental operations of marginalising, updating and renewing (*update()*, *continuous_marg()*, *discrete_marg()*). The current probability distribution for a discrete node or marginal mean and variance for a continuous node can easily be derived from these cliques or separators.

collectevidence()

collect evidence is used when evidence from the entire clique tree must propagate to a single clique

The root clique issues a message "collect evidence" to its neighbours, who pass it to their neighbours until it reaches the leaves of the tree. Each clique then collects evidence from those neighbours further from the

root clique, using the fundamental operations: marginalisation, update and renew.

distributeevidence()

distribute evidence is used when evidence from a single clique must propagate to the entire clique tree

In the procedure of distribution, the same message passing operation as evidence collection is involved but working back through the tree.

update()

update CG potentials on the clique

The calculation involves the clique and its neighbours with their separators S. The potentials are modified by multiplying each term by the associated update ratio, which is the ratio of the relevant value of the new potential over S to that of the old one.

node_marginal()

find out marginal probabilities of discrete nodes, means and variances of continuous nodes

The marginal probabilities of discrete nodes, means and variances of continuous nodes can be obtained by using two procedures *discrete_marg()* and *continuous_marg()* on the clique containing the node of interest. The information is placed in the data structure *node_marg*, which will be accessed by the procedure *node_display* to present to the user.

node_distribution()

display marginal probability distributions of the nodes in the graph

Marginal probabilities of discrete nodes, means and variances of continuous variables are displayed.

planning()

order questions concerning a node of interesting

influence_find()

find the most influential evidence for a belief

simulation()

perform stochastic simulation

activatenodes()

simulate a value for each node

The conditional probabilities of a node given all other states are calculated. A new state of the node is determined based on the conditional probabilities.

REFERENCES

1. Buchanan, B.G. and Shortliffe, E.H., *Rule-based Expert Systems: the MYCIN Experiment of the Stanford Heuristic Programming Project*, Reading: Addison-Wesley, (1984).

2. Duda, R.O., Gaschning, J., and Hart, P.E., "Model Design in the Prospector Consultant System for Mineral Exploration," *Expert Systems in the Micro-electronic Age*, pp. 153-167, Edinburgh University Press, (1979).

3. Miller, R.A., Pople, H.E. Jr, and Myers, J.D., "INTERNIST-1, An Experimental Computer-based Diagnostic Consultant for General Internal Medicine," *New England Journal of Medicine*, **Vol. 307**, pp. 468-476, (1982).

4. Dombal, F.T.de, Leaper, D.J., Staniland, J.R., McCann, A.P., and Horrocks, J.C., "Computer-aided Diagnosis of Acute Abdominal Pain," *British Medical Journal*, **Vol. 2**, pp. 9-13, (1972).

5. Adams, J.B., "A Probability Model of Medical Reasoning and the MYCIN Model," *Mathematical Bioscience*, **Vol. 32**, pp. 177-186, American Elsevier Publishing Company, Inc., (1976).

6. Zadeh, L.A., "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems," *Fuzzy Sets and Systems*, **Vol. 11**, pp. 199-228, (1983).

7. Spiegelhalter, D.J. and Knill-Jones, R.P., "Statistical and Knowledge-based Approaches to Clinical Decision-support Systems with an Application in Gastro-enterology (with discussion)," *Journal of the Royal Statistical Society (A)*, **Vol. 147**, pp. 35-77, (1984).

8. Bonissone, P.P. and Tong, R.M., "Editorial: Reasoning with Uncertainty in Expert Systems," *Int. J. Man-machine Studies*, **Vol. 22**, pp. 241-250, Academic Press, (1985).

9. Cohen, P.R., *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach*, Pitman, Boston, (1985).

10. Prade, H., "A Computational Approach to Approximate and Plausible Reasoning with Applications to Expert Systems," *IEEE PAMI*, **Vol. 7**, pp. 260-283, (1985).

11. Cohen, P.R., "The Control of Reasoning Under Uncertainty: A Discussion of Some Programs," *The knowledge Engineering Review*, **Vol. 2**, pp. 5-25, (1987).

12. Duda, R.O., Hart, P.E., and Nilsson, N.J., "Subjective Bayesian Methods for Rule-based Inference Systems," *Proc. 1976 National Computer Conference (AFIPS)*, **Vol. 45**, pp. 1075-1082, (1976).

13. Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, N.J., (1976).

14. Quinlan, J.R., "INFERNO: A Cautious Approach to Uncertain Inference," *The Computer Journal*, **Vol. 26**, pp. 255-269, (1983).

15. Bundy, A., "Incidence Calculus: A Mechanism for Probabilistic Reasoning," *Journal of Automated Reasoning*, **Vol. 1**, pp. 263-284, (1985).

16. Nilsson, N.J., "Probabilistic Logic," *Artificial Intelligence*, **Vol. 28**, pp. 71-87, (1986).

17. Henkind, S.J. and Harrison, M.C., "An Analysis of Four Uncertainty Calculi," *IEEE Trans. on Systems, Man, and Cybernetics*, **Vol. 18**, pp. 700-714, (1988).

18. Liu, X., Zhang, G., and Luo, Z., "A Methodological Approach to Visual Reasoning," *Proc. of International Workshop on Industrial Applications of Machine Intelligence and Vision*, pp. 50-55, Tokyo, Japan, (April 1989).

19. Moore, E.C., "Semantic Considerations on Non-Monotonic Logic," *Artificial Intelligence*, Vol. 25, (1985).

20. Zadeh, L.A., "Is Probability Theory Sufficient for Dealing With Uncertainty in AI: a Negative View," *Uncertainty in Artificial Intelligence, L.N.Kanal and J.F.Lemmer(Editors)*, pp. 103-116, (1986).

21. Shortliffe, E.H. and Buchanan, B.G., "A Model of Inexact Reasoning in Medicine," *Math. Biosci.*, Vol. 23, pp. 351-379, (1975).

22. Gammerman, A. and Thatcher, A.R., "Bayesian Diagnostic Probabilities without Assuming Independence of Symptoms," *Methods of Information in Medicine*, Vol. 30, pp. 15-22, E.K. Schattauer Verlagsgesellschaft mbH, (1991).

23. Dempster, A.P., "Upper and Lower Probabilities Induced by a Multivalued Mapping," *Annals Mathematics Statistics*, Vol. 38, pp. 325-339, (1967).

24. Howard, R.A. and Matheson, J.E., "Influence Diagrams," *Readings on the Principles and Applications of Decision Analysis (R.A. Howard and J.E. Matheson, editors)*, Vol. 2, pp. 721-762, Strategic Decision Group, Menlo Park, CA, (1981).

25. Pearl, J., "Fusion, Propagation, and Structuring in Belief Networks," *Artificial Intelligence*, Vol. 29, pp. 241-288, (1986).

26. Spiegelhalter, D.J., "Probabilistic Reasoning in Predictive Expert Systems," *Uncertainty in Artificial Intelligence, L.N.Kanal and J.F.Lemmer(Editors)*, pp. 47-67, (1986).

27. Shachter, R.D., "Evaluating Influence Diagrams," *Operations Research*, Vol. 34, pp. 871-882, (1986).

28. Lauritzen, S.L. and Spiegelhalter, D.J., "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems (with discussion)," *Journal of the Royal Statistical Society (B)*, Vol. 50, pp. 157-224, (1988).

29. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, INC., San Mateo, California, (1988).

30. Jensen, F.V., Lauritzen, S.L., and Olesen, K.G., "Bayesian Updating in Causal Probabilistic Networks by Local Computations," *Computational Statistics Quarterly*, Vol. 4, pp. 269-282, John Wiley & Sons, Inc., (1990).

31. Jensen, F.V., Olesen, K.G., and Andersen, S.K., "An Algebra of Bayesian Belief Universes for Knowledge Based Systems," *Networks*, Vol. 20, pp. 637-659, John Wiley & Sons, Inc., (1990).

32. Pearl, J., "Evidential Reasoning Using Stochastic Simulation of Causal Models," *Artificial Intelligence*, Vol. 32, pp. 245-257, (1987).

33. Luo, Z. and Gammerman, A., "PRESS - A Probabilistic Reasoning Expert System Shell," *European Conference on Symbolic and Quantitative Approaches for Uncertainty, also in Lecture Notes in Computer Science 548 (R.Kruse and P.Siegel Eds.)*, pp. 232-237, Springer-Verlag, Marseille, France, (Oct. 1991).

34. Lauritzen, S.L., "Propagation of Probabilities, Means and Variances in Mixed Graphical Association Models," *Journal of the American Statistical Association*, Vol. 86, (1992).

35. Gammerman, A., Aitken, C.G.G., Luo, Z., and Talbot, M., "A Computational System for Mixed Probabilistic Models," *Submitted to the 8th Conference on Uncertainty in Artificial Intelligence*, (1992).

36. Spiegelhalter, D.J. and Lauritzen, S.L., "Sequential Updating of Conditional Probabilities on Directed Graphical Structures," *Networks*, **Vol. 20**, (5), pp. 579-605, John Wiley & Sons, Inc.. (1990).

37. Gammerman, A. and Luo, Z., "Constructing Causal Trees From a Medical Database," *Technical Report TR91002*, Dept. of Computer Science, Heriot-Watt University, Edinburgh, UK, (1991).

38. Gammerman, A. and Thatcher, A.R., "Bayesian Inference in an Expert System without Assuming Independence," *Advances in Artificial Intelligence (ed. M. Golumbic)*, pp. 182-219, Springer-Verlag, (1990).

39. Savage, L.J., *The Foundation of Statistics*, New York: Wiley (2rd revised ed. 1972 Dover.), (1954).

40. Finetti, B. de, *Theory of Probability*, Wiley, New York, (1974).

41. Lindley, D.V., "Scoring Rules and the Inevitability of Probability," *International Statistical Review*, **Vol. 50**, pp. 1-26, (1982).

42. Reboh, R., "Knowledge Engineering Techniques and Tools in the PROSPECTOR Environment," *Technical Note 243*, SRI International, (1981).

43. Staniland, J.R., Ditchburn, J., and Dombal, F.T. de, "Clinical Presentation of Acute Abdomen Study of 600 Patients," *British Medical Journal*, **Vol. 3**, pp. 393-398, (1972).

44. Dombal, F.T.de, "Surgical Diagnosis Assisted by Computer," *Proc. R. Soc. Lond. B* , **Vol. 184**, pp. 433-440, (1973).

45. Dombal, F.T.de, Leaper, D.J., and Horrocks, J.C., "Human and Computer Aided Diagnosis of Acute Abdominal Pain: Future Report with Emphasis on the Performance of Clinicians," *British Medical Journal*, **Vol. 1**, pp. 376-380, (1974).

46. Heckerman, D.E., Horvitz, E.J., and Nathwani, B.N., "Toward Normative Expert Systems: The PATHFINDER Project," *Methods of Information in Medicine*, Forthcoming, (1991).

47. Heckerman, D., "An Empirical Comparison of Three Inference Methods," *Proc. of 4th Workshop on Uncertainty in AI*, pp. 158-169, (August 1988).

48. Heckerman, D., "Probabilistic Similarity Networks," *Networks*, **Vol. 20**, pp. 607-636, John Wiley & Sons, Inc., (1990).

49. Cumberbatch, J. and Heaps, H.S., "A Diseaseconscious Method for Sequential Diagnosis by Use of Disease Probabilities without Assumption of Symptom Independence," *Int. J. Biomed. Comp.*, **Vol. 7**, pp. 66-78, (1976).

50. Geman, S., "Stochastic Relaxation Methods for Images Restoration and Expert Systems," *in Maximum-Entropy and Bayesian Methods in Science and Engineering (Edited by G.J. Erickson and C.R. Smith)*, **Vol. 2**, pp. 265-311, (1988).

51. Thatcher, A.R., "Computer Models of Probabilistic Reasoning: Bayes' Theorem without Assuming Independence," *Technical Report No. 88/1*, Dept. of Computer Science, Heriot-Watt University, Edinburgh, (1988).

52. Thatcher, A.R., "Relationships between Bayesian and Confidence Limits for Predictions," *Journal of the Royal Statistical Society (B)*, **Vol. 26**, pp. 176-192, (1964).

53. Quinlan, J.R., "Learning Efficient Classification Procedures and Their Application to Chess End Games," *Machine Learning: An Artificial Intelligence Approach, Edited by R.S. Michalski et. al.*, pp. 463-482, Tioga Publ. Co., Palo Alto, (1983).

54. Quinlan, J.R., "Induction of Decision Trees," *Machine Learning, 1*, pp. 81-106, (1986).

55. Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J., *Classification and Regression Trees*, Wadsworth Int. Group, Belmont, California, (1984).

56. Andersen, S.K., Olesen, K.G., Jensen, F.V., and Jensen, F., "HUGIN - A Shell for Building Bayesian Belief Universes for Expert Systems," *Proc. of 11th IJCAI*, Vol. 2, pp. 1080-1085, (1989).

57. Chavez, R.M. and Cooper, G.F., "A Randomized Approximation Algorithm for Probabilistic Inference on Bayesian Belief Networks," *Networks*, Vol. 20, pp. 661-685, John Wiley & Sons, Inc., (1990).

58. Andreassen, S., Woldbye, M., Falck, B., and Andersen, S.K., "MUNIN - A Causal Probabilistic Network for Interpretation of Electromyographic Findings," *Proc. of 10th IJCAI*, Vol. 2, pp. 366-372, Milan, (Aug. 1987).

59. Cooper, G.F., *NESTOR: A Computer-based Medical Diagnostic Aid that Integrates Causal and Probabilistic Knowledge*, Ph.D. dissertation, Medical Information Science, Stanford University, (1984).

60. Wen, W.X., "Parallel MCE Reasoning and Boltzmann-Jeffery Machine Networks," *Proc. 3nd IEEE Symp. on Intelligent Control*, (1988).

61. Miller, A.C., Merkhofer, M.W., Howard, R.A., Matheson, J.E., and Rice, T.R., *Development of Automated Aids for Decision Analysis*, Stanford Research Institute, Menlo Park, Calif., (1976).

62. Olmsted, S.M., *On Representing and Solving Decision Problems*, Ph.D. Thesis, Engineering-Economic Systems Dept., Stanford University, Stanford, Calif., (1983).

63. Wellman, M.P., "Qualitative Probabilistic Networks for Planning Under Uncertainty," *Uncertainty in Artificial Intelligence*, Vol. 2, pp. 197-207, Elsevier Science Publishers B.V. (North-Holland), (1988).

64. Wellman, M.P., "Graphical Inference in Qualitative Probabilistic Networks," *Networks*, Vol. 20, pp. 687-701, John Wiley & Sons, Inc., (1990).

65. Fertig, K.W. and Breese, J.S., "Interval Influence Diagrams," *Uncertainty in Artificial Intelligence 5 (ed. M. Henrion, R.D: Shachter, L.N. Kanal, and J.F. Lemmer)*, pp. 149-161, Elsevier Science Publishers, (1990).

66. Shenoy, P.P. and Shafer, G., "Propagating Belief Functions with Local Computations," *IEEE Expert*, pp. 43-51, (Fall 1986).

67. Shenoy, P.P. and Shafer, G., "An Axiomatic Framework for Bayesian and Belief-function Propagation," *Proc. of 4th Workshop on Uncertainty in AI*, pp. 307-314, (August 1988).

68. Shenoy, P.P., Shafer, G., and Mellouli, K., "Propagation of Belief Functions: A Distributed Approach," *Uncertainty in Artificial Intelligence*, Vol. 2, pp. 325-335, Elsevier Science Publishers B.V., North-Holland, (1988).

69. Dempster, A.P. and Kong, A., "Uncertain Evidence and Artificial Analysis," *Journal of Statistical Planning and Inference*, Vol. 20, pp. 355-368, North-Holland, (1988).

70. Kim, J.H. and Pearl, J., "A Computational Model for Causal and Diagnostic Reasoning in Inference Systems," *Proc. of 8th IJCAI*, Vol. 1, pp. 190-193, Karlsruhe, W.G., (1983).

71. Henrion, M., "Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling," *Uncertainty in Artificial Intelligence 2*, pp. 149-163, (1988).

72. Suermondt, H.J. and Cooper, G.F., "Updating Probabilities in Multiply-connected Belief Networks," *Proc. of 4th Workshop on Uncertainty in AI*, pp. 335-343, (August 1988).

73. Suermondt, H.J. and Cooper, G.F., "Initialization for the Method of Conditioning in Bayesian Belief Networks," *Artificial Intelligence*, Vol. 50, pp. 83-94, Elsevier Science Publishers B.V., (1991).

74. Darroch, J.N., Lauritzen, S.L., and Speed, T.P., "Markov Fields and Log-linear Models for Contingency Tables," *Ann. Statist.*, Vol. 8, pp. 522-539, (1980).

75. Lauritzen, S.L., Speed, T.P., and Vijayan, K., "Decomposable Graphs and Hypergraphs," *J. Austral. Math. Soc.(Series A)*, Vol. 36, pp. 12-29, (1984).

76. Lauritzen, S.L., Dawid, A.P., Larsen, B.N., and Leimer, H.-G., "Independence Properties of Directed Markov Fields," *Networks*, Vol. 20, pp. 491-505, John Wiley & Sons, Inc., (1990).

77. Henrion, M., "Towards Efficient Probabilistic Diagnosis in Multiply Connected Belief Networks," *Influence Diagrams, Belief Nets and Decision Analysis (eds R.M. Oliver & J.Q. Smith)*, pp. 385-410, Wiley, Chichester, (1990).

78. Kjaerulff, U., "Triangulation of Graphs - Algorithms Giving Small Total State Space," *Research Report R 90-09*, Institute for Electronic Systems, Aalborg University, Denmark, (March 1990).

79. Shwe, M., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., and Lehmann, H., "Probabilistic Diagnosis Using a Reformulation of the INTERNIST-1/QMR Knowledge Base: I. The Probabilistic Model and Inference Algorithms," *Methods of Information in Medicine*, Vol. 30, pp. 241-255, (1991).

80. Cooper, G.F., "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks," *Artificial Intelligence*, Vol. 42, pp. 393-405, (1990).

81. Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, (1979).

82. Geman, S. and Geman, D., "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 6, pp. 721-741, (November 1984).

83. Fung, R. and Chang, K.-C., "Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks," *Uncertainty in Artificial Intelligence 5 (ed. M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer)*, pp. 209-219, Elsevier Science Publishers, (1990).

84. Shachter, R.D. and Peot, M.A., "Simulation Approaches to General Probabilistic Inference on Belief Networks," *Uncertainty in Artificial Intelligence 5 (ed. M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer)*, pp. 221-231, Elsevier Science Publishers, (1990).

85. Chavez, R.M. and Cooper, G.F., "An Empirical Evaluation of a Randomized Algorithm for Probabilistic Inference," *Uncertainty in Artificial Intelligence 5 (ed. M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer)*, pp. 191-207, Elsevier Science Publishers, (1990).

86. Hrycej, T., "Gibbs Sampling in Bayesian Networks," *Artificial Intelligence*, Vol. 46, pp. 351-363, Elserier Science Publishers (North-Holland), (1990).

87. Luo, Z. and Gammerman, A., "A Stochastic Simulation System and Its Application to Causal Models," *3rd International Conference on IPMU in Knowledge-Based Systems*, pp. 186-89, Paris, France, (July 1990).

88. Luo, Z. and Gammerman, A., "STOSS - A Stochastic Simulation System for Bayesian Belief Networks," *Lecture Notes in Computer Science (edited B. Bouchon-Meunier R.R. Yager L.A. Zadeh)*, Vol. 521, pp. 97-105, Springer-Verlag, (1991).

89. Chin, H.L. and Cooper, G.F., "Bayesian Belief Network Inference Using Simulation," *Uncertainty in Artificial Intelligence 3*, North-Holland, Amsterdam, (1989).

90. Shachter, R.D., "Probabilistic Inference," *Operations Research*, Vol. 36, pp. 589-604, (1988).

91. Shachter, R.D. and Heckerman, D.E., "A Backwards View for Assessment," *Uncertainty in Artificial Intelligence*, Vol. 2, pp. 317-323, Elsevier Science Publishers B.V., North-Holland, (1988).

92. Kong, A., "Multivariate Belief Functions and Graphical Models," *Ph.D. Thesis*, Department of Statistics, Harvard University, (1986).

93. Zhang, L., "Studies on Finding Hypertree Covers for Hypergraphs," *School of Business Working Paper No. 198*, University of Kansas, (1988).

94. McCarthy, J., "Circumscription - A Form of Non-monotonic Reasoning," *Artificial Intelligence*, Vol. 13, pp. 27-39; 171-172, (1980).

95. Cooper, G.F., "Current Research Directions in the Development of Expert Systems Based on Belief Networks," *Applied Stochastic Models and Data Analysis*, Vol. 5, pp. 39-52, (1989).

96. Shachter, R.D., "DAVID: Influence Diagram Processing System for the Macintosh," *In Uncertainty in Artificial Intelligence 2*, pp. 191-196, North-Holland, Amsterdam, (1988).

97. Aitken, C.G.G. In discussion of Lauritzen, S.L. and Spiegelhalter, D.J., "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems." *Journal of the Royal Statistical Society (B)*, Vol. 50, pp. 200-201, (1988).

98. Chavez, R.M. and Cooper, G.F., "KNET: Integrating Hypermedia and Bayesian Modeling," *Proc. of 4th Workshop on Uncertainty in AI*, pp. 49-54, (August 1988).

99. Zarley, D., Hsia, Y., and Shafer, G., "Evidential Reasoning Using DELIEF," *Proc. of AAAI-88*, Vol. 1, pp. 205-209, (1988).

100. Beinlich, I. and Herskovits, E., "ERGO: A Graphical Environment for Constructing Bayesian Belief Networks," *Proceedings of the Sixth Conference on Uncertainty in Artifical Intelligence*, pp. 114-121, Cambridge, Massachusset, (1990).

101. Cooper, G.F., "A Method for Using Belief Networks as Influence Diagrams," *Proc. of 4th Workshop on Uncertainty in AI*, pp. 55-63, (August 1988).

102. Shafer, G., Shenoy, P.P., and Mellouli, K., "Propagation of Belief Functions in Qualitative Markov Tree," *International Journal of Approximate Reasoning*, Vol. 1, pp. 349-400, (1987).

103. Kullback, S., *Information Theory and Statistics*, Dover Publications, New York, (1968).

104. Luo, Z. and Gammerman, A., "A Probabilistic Reasoning System Based on a Causal Graph Approach," *IEE Colloquium on "Knowledge Engineering"*, pp. 4/1-4/3, London, U.K., (May 1990).

105. Grieve, M.C., Dunlop, J., and Haddock, P.S., "Transfer Experiments with Acrylic Fibres," *Forensic Science International*, Vol. 40, pp. 267-277, (1989).

106. Shachter, R.D. and Kenley, C.R., "Gaussian Inference Diagrams," *Management Science*, Vol. 35, pp. 527-550, (May 1989).

107. Bellazzi, R., Berzuini, C., Quaglini, S., Spiegelhalter, D.J., and Leaning, M., "Cytoxic Chemotherapy Monitoring Using Stochastic Simulation on Graphical Models," *Proc. of AIME 91 Conference*, Maastricht, The Netherlands, (June 1991).

108. Andreassen, S., Hovorka, R., Benn, J., Olesen, K.G., and Carson, E., "A Model-based Approach to Insulin Adjustment," *Proc. of AIME 91 Conference*, Maastricht, The Netherlands, (June 1991).

109. Lauritzen, S.L. and Wermuth, N., "Graphical Models for Associations Between Variables, Some of Which Are Qualitative and Some Quantitative." *The Annals of Statistics*, Vol. 17, pp. 31-57, (1989).

110. Lauritzen, S.L., "Mixed Graphical Association Models (with discussion)." *Scand. J. Statist.*, Vol. 16, pp. 273-306, (1989).

111. Frydenberg, M. and Lauritzen, S.L., "Decomposition of Maximum Likelihood in Mixed Graphical Interaction Models." *Biometrika*, Vol. 76, pp. 539-555, (1989).

112. Wermuth, N. and Lauritzen, S.L., "On Substantive Research Hypotheses, Conditional Independence Graphs and Graphical Chain Models (with discussion)." *Journal of the Royal Statistical Society (B)*, Vol. 52, pp. 21-72, (1990).

113. Leimer, H.-G., "Triangulated Graphs with Marked Vertices," *Annals of Discrete Mathematics*, Vol. 41, pp. 311-324, (1989).

114. Normand, S.L. and Tritchler, D., "Computation in Multiprocess Dynamic Linear Models: A Network Approach," *Proceedings of the Statistical Computing Section, American Statistical Association*, (1991).

115. Brewer, M.J., Aitken, C.G.G., Luo, Z., and Gammerman, A., "Stochastic Simulation in Mixed Graphical Association Models," *Technical Report*, Dept. of Statistics, Edinburgh University, (1992).

116. Kiiveri, H., , T.P. Speed, and Carlin, J.B., "Recursive Causal Models," *J. Austral. Math. Soc.(Series A)*, Vol. 36, pp. 30-52, (1984).

117. Tversky, A. and Kahneman, D., "Judgement Under Uncertainty: Heuristics and Biases," *Science*, Vol. 185, pp. 1124-31, (Sept. 1974).

118. Kahneman, D., , P. Slovic, and Tversky, A., *Judgments Under Uncertainty: Heuristics and Biases*, Cambridge University Press, Cambridge, (1982).

119. Duda, R.O. and Hart, P.E., *Pattern Classification and Scene Analysis*, Wiley, New York, (1973).

120. Titterington, D.M., "Updating a Diagnostic System Using Unconfirmed Cases," *Appl. Statist.*, Vol. 25, pp. 238-247, (1976).

121. Smith, A.F.M. and Makov, U.E., "A Quasi-Bayes Sequential Procedure for Mixtures ," *Journal of the Royal Statistical Society (B)*, Vol. 40, pp. 106-111, (1978).

122. Gammerman, A. and Crabbe, W., "Computational Models of Probabilistic Reasoning in Expert Systems: A Causal Probabilistic Reasoning System." *Technical Report 87/16. Computer Science Department, Heriot-Watt University, Edinburgh.*, (1987).

123. Gammerman, A., "A Causal Probabilistic Reasoning System," *Proc. of 4th International Symposium on Knowledge Engineering*, pp. 391-399, Barcelona, (May 1990).

124. Herskovits, E. and Cooper, G., "Kutato: An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases," *Report KSL-90-22*, Medical Computer Science, Stanford University, (March 1990).

125. Srinivas, S., Russell, S., and Agogino, A., "Automated Construction of Sparse Bayesian Networks from Unstructured Probabilistic Models and Domain Information," *Uncertainty in Artificial Intelligence 5 (ed. M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer)*, pp. 295-308, Elsevier Science Publishers, (1990).

126. Hilary, F. and Palmer, E., *Graphical Enumeration*, Academic Press, New York, (1973).

127. Chow, C.K. and Liu, C.N., "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Trans. on Info. Theory*, **Vol. IT-14**, pp. 462-467, (1968).

128. Spiegelhalter, D.J. and Smith, A.F.M., "Bayes Factors for Linear and Log-linear Models with Vague Prior Information," *Journal of the Royal Statistical Society (B)*, **Vol. 44**, pp. 377-387, (1982).

129. Spiegelhalter, D.J. and Lauritzen, S.L., "Techniques for Bayesian Analysis in Expert Systems," *Annals of Mathematics and Artificial Intelligence* , **Vol. 2**, pp. 353-366, (1990).

130. Luo, Z. and Gammerman, A., "Structure Learning in Bayesian Belief Networks," *Submitted to IEEE Transaction on Pattern Analysis and Machine Intelligence*, (1991).

131. Feigenbaum, E.A., "Themes and Case Studies of Knowledge Engineering," *Expert Systems in the Micro Electronic Age, D.Michie (editor)*, pp. 3-25, Edinburgh University Press, Edinburgh, (1979).

132. Berzuini, C., "Modeling Temporal Processes via Belief Networks and Petri Nets, with Application to Expert Systems," *Annals of Mathematics and Artificial Intelligence*, **Vol. 2**, pp. 39-64, (1990).

133. Verma, T. and Pearl, J., "Equivalence and Synthesis of Causal Models," *Proceedings 6th Conference on Uncertainty in AI*, pp. 220-227, (1990).

134. Pearl, J. and Verma, T.S., "A Theory of Inferred Causation," *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pp. 441-452, Morgan Kaufmann, San Mateo, CA, (April 1991).

135. Reichenbach, H., *The Direction of Time*, University of California Press, Berkely, (1956).

136. Good, I.J., "A Causal Calculus," *British Journal for Philosophy of Science*, **Vol. 11,12,13**, pp. 305-328,43-51,88, (1983).

137. Wallace, C.S. and Freeman, P.R., "Estimation and Inference by Compact Coding," *Journal of the Royal Statistical Society (B)*, **Vol. 49**, pp. 240-265, (1987).

138. Rissanen, J., "Modelling by Shortest Data Description," *Automatica*, **Vol. 14**, pp. 465-471, (1978).

139. Rose, D.J., "A Graph-theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations," *Graph Theory and Computing (R. Read, ed.)*, pp. 183-217, Academic Press, New York, (1973).

140. Cannings, C., Thompson, E.A., and Skolnick, M.H., "Recursive Derivation of Likelihoods on Pedigrees of Arbitrary Complexity," *Adv. Appl. Probabil.*, **Vol. 8**, pp. 622-625, (1976).

141. Cannings, C., Thompson, E.A., and Skolnick, M.H., "Probability Functions on Complex Pedigrees," *Adv. Appl. Probabil.*, Vol. 10, pp. 26-61, (1978).

142. Berge, C., *Graph and Hypergraph*, North-Holland Publishing Co., (1973).

143. Golombic, M.C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, London, (1980).

144. Dirac, G.A., "On Rigid Circuit Graphs," *Abh. Math. Sem. Univ. Hamburg*, Vol. 25, pp. 71-76, (1961).

145. Gavril, F., "An Algorithm for Testing Chordality of Graphs," *Information Precessing Letters*, Vol. 3, pp. 110-112, (1974).

146. Tarjan, R.E. and Yannakakis, M., "Simple Linear-time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs and Selectively Reduce Acyclic Hypergraphs," *SIAM J.Computer*, Vol. 13, pp. 566-579, (Aug. 1984).

147. Rose, D.J., Tarjan, R.E., and Lueker, G.S., "Algorithmic Aspects of Vertex Elimination on Graphs," *SIAM J. Computer*, Vol. 5, pp. 266-283, (June 1976).

148. Yannakakis, M., "Computing the Minimum Fill-in Is NP-complete," *SIAM J. Alg. Disc. Meth.*, Vol. 2, pp. 77-79, (1981).

149. Isham, V., "An Introduction to Spatial Point Processes and Markov Random Fields," *International Statistical Review*, Vol. 49, pp. 21-43, (1981).

150. Dawid, A.P., "Conditional Independence in Statistical Theory," *Journal of the Royal Statistical Society (B)*, Vol. 41, pp. 1-31, (1979).

151. Smith, J.Q., *Decision Analysis: A Bayesian Approach*, Chapman & Hall, London, (1987).

152. Geiger, D. and Pearl, J., "On the Logic of Causal Models," *Proc. of 4th Workshop on Uncertainty in AI*, pp. 136-147, (August 1988).

153. Geiger, D., Verma, T., and Pearl, J., "d-separation: From Theorems to Algorithms," *Uncertainty in Artificial Intelligence 5 (ed. M. Henrion, R.D. Shachter, L.N. Kanal, and J.F. Lemmer)*, pp. 139-48, Elsevier Science Publishers, (1990).

154. Geiger, D., Verma, T., and Pearl, J., "Identifying Independence in Bayesian Networks," *Networks*, Vol. 20, pp. 507-534, (1990).

155. Pearl, J. and Paz, A., "Graphoids: A Graph-based Logic for Reasoning about Relevancy Relations," *Proceedings of the European Conference on Artificial Intelligence*, Brighton, UK, (1986).

156. Pearl, J. and Verma, T., "The Logic of Representing Dependencies by Directed Graphs," *AAAI 87*, pp. 374-379, (1987).

157. Smith, J.Q., "Influence Diagrams for Statistical Modelling," *Annals of Statistics*, Vol. 17, pp. 654-672, (1989).

158. Wilson, J.R., "Statistical Aspects of Simulation," *Operational Research*, pp. 921-937, (1984).

159. Law, A.M. and Kelton, W.D., *Simulation Modeling and Analysis*, McGraw-Hill, New York, (1982).

160. Wilson, J.R., "Variance Reduction Techniques For Digital Simulation," *American Journal of Mathematical and Management Sciences*, Vol. 4, pp. 277-312, (1984).

161. Anderson, T.W., "An Introduction to Multivariate Statistical Analysis," *(Second Edition)*, John Wiley & Sons, (1984).